

Information – Organisation – Produktion

Hrsg.: Hans Corsten, Michael Reiß, Claus Steinle, Stephan Zelewski

Lars Uwe Dittmann

OntoFMEA

Ontologiebasierte Fehlermöglichkeits- und Einflussanalyse



GABLER EDITION WISSENSCHAFT

Lars Uwe Dittmann

OntoFMEA

GABLER EDITION WISSENSCHAFT

Information – Organisation – Produktion

Herausgegeben von Professor Dr. Hans Corsten,
Professor Dr. Michael Reiß, Professor Dr. Claus Steinle
und Professor Dr. Stephan Zelewski

Die Schriftenreihe präsentiert Konzepte, Modelle und Methoden zu drei zentralen Domänen der Unternehmensführung. Information, Organisation und Produktion werden als Bausteine eines integriert angelegten Managementsystems verstanden. Der Erforschung dieses Bereiches dienen sowohl theoretische als auch anwendungsorientierte Beiträge.

Lars Uwe Dittmann

OntoFMEA

Ontologiebasierte Fehlermöglichkeits- und Einflussanalyse

Mit einem Geleitwort von Univ.-Prof. Dr. Stephan Zelewski

Deutscher Universitäts-Verlag

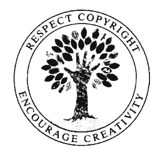
Bibliografische Information Der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der
Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über
<http://dnb.d-nb.de> abrufbar.

Dissertation Universität Duisburg-Essen, Campus Essen, 2006

1. Auflage Juni 2007

Lektorat: Frauke Schindler / Nicole Schweitzer

Der Deutsche Universitäts-Verlag ist ein Unternehmen von Springer Science+Business Media.
www.duv.de



Umschlaggestaltung: Regine Zimmer, Dipl.-Designerin, Frankfurt/Main
Gedruckt auf säurefreiem und chlorfrei gebleichtem Papier
Printed in Germany

ISBN 978-3-8350-0749-9

Geleitwort

Das Werk von Herrn Dr. Dittmann befasst sich mit einer komplexen, interdisziplinär ausgerichteten Problemstellung an der Nahtstelle zwischen Betriebswirtschaftslehre, Wirtschaftsinformatik und Informatik. Es befasst sich mit der Integration zweier Themengebiete, die bislang weit gehend voneinander isoliert behandelt wurden. Auf der einen Seite wird aus dem breiten Themenfeld des Qualitätsmanagements die spezielle Technik der Failure Mode and Effects Analysis (FMEA) behandelt. Sie gehört neben einigen anderen Exemplaren, wie etwa dem Quality Function Deployment, zu den bekanntesten und inhaltlich anspruchsvollsten Analysetechniken des „modernen“ Qualitätsmanagements. Auf der anderen Seite widmet sich der Autor dem computergestützten Wissensmanagement. Aus diesem ebenso facettenreichen Themenfeld hat er sich die spezielle Variante ontologiebasierter Wissensmanagementsysteme herausgegriffen. Ontologien spielen sowohl im Rahmen des so genannten Knowledge Level Engineerings (Newell) als auch des Semantic Webs (Berners-Lee) eine herausragende Rolle bei der Vermittlung zwischen Informatik und Erforschung Künstlicher Intelligenz einerseits sowie Fachdisziplinen wie der Betriebswirtschaftslehre andererseits. Mit seinem ambitiösen Forschungsprojekt, die beiden nur scheinbar unverbundenen Themengebiete der FMEA-Technik sowie ontologiebasierter Wissensmanagementsysteme zu einer fruchtbaren Synthese zusammenzuführen, hat sich der Autor an der „vorderen Frontlinie“ betriebswirtschaftlicher Forschung und ihrer Umsetzung in die betriebliche Praxis positioniert.

Im Zentrum der hier vorgelegten Forschungsresultate steht die Erstellung von FMEA-Analysen mit Hilfe von Ontologien und darauf aufbauenden, ontologiebasierten Wissensmanagementsystemen. Dabei hat der Autor stets großen Wert auf die *Anschlussfähigkeit* seiner Untersuchungen zur *betrieblichen Praxis* gelegt. Dies manifestiert sich u.a. darin, dass seine Überlegungen oftmals an die aktualisierte Norm DIN EN ISO 9000:2000 anknüpfen, die für die Wirtschaftspraxis im Rahmen von Auditierungen und Zertifizierungen eine herausragende Rolle spielt. Darüber hinaus demonstriert der Autor die unmittelbare Praxisrelevanz seiner Forschungsarbeiten auch anhand zweier Praxisbeispiele, die er in enger Kooperation mit einem Unternehmen aus der Anlagen- und Maschinenbaubranche, der Kölner Karl Schumacher Maschinenbau (KSM) GmbH, entwickelt hat. Es handelt sich um zwei Spezialmaschinen, einen „5-Automatik-Spindelschrauber“ und einen „9+1-Spindelschrauber“, die von der KSM GmbH für die Montage von Automobilgetrieben produziert werden. Anhand dieser beiden Beispiele gelingt es dem Autor in bemerkenswerter Weise, die zunächst sehr „theoretisch“ erscheinende und formalsprachlich hoch anspruchsvolle Technik der Wissensmodellierung mit Hilfe von Ontologien so aufzubereiten, dass sich ihre grundsätzliche Bedeutung und ihre „ökonomische Verwertbarkeit“ für die Wirtschaftspraxis unmittelbar nachvollziehen lassen.

Um die Erstellung von ontologiebasierten Wissensmanagementsystemen konkret zu unterstützen, hat sich der Autor intensiv mit *Vorgehensmodellen* aus den drei Bereichen des Software Engineerings, des Knowledge Engineerings und des Ontology Engineerings auseinander gesetzt. Auf dieser breiten Grundlage hat er ein eigenes OntoFMEA-Vorgehensmodell entwickelt, das speziell auf die Konstruktion von Ontologien für Zwecke des betrieblichen Qualitätsmanagements (Domäne) unter Einsatz der FMEA-Technik (Methode) zugeschnitten ist. Mit einem anerkennenswerten Spürsinn für einen Kompromiss zwischen Anwendungsnähe

und Praktikabilität einerseits sowie wissenschaftlichem Anspruch andererseits ist es dem Autor gelungen, ein neuartiges Vorgehensmodell für die vorgenannte Domänen-Methoden-Kombination zu entwickeln, für das in der einschlägigen Fachliteratur kein Pendant existiert. In dieser Hinsicht ist ihm eine *konzeptionelle Innovation* gelungen. Sie wird hoffentlich eine rasche Verbreitung und eine konstruktiv-kritische Rezeption unter Lesern finden, die sich in den Bereichen Qualitätsmanagement, FMEA-Technik sowie ontologiebasierte Wissensmanagementsysteme engagieren.

Der Autor demonstriert die Anwendbarkeit seines OntoFMEA-Vorgehensmodells anhand der systematischen Erstellung einer konkreten Ontologie für die FMEA-Technik. Mit dieser detailliert ausgearbeiteten FMEA-Ontologie wird der aktuelle State-of-the-art in den Bereichen FMEA-Technik und Ontologien deutlich übertroffen. Zwar existieren in der Fachliteratur einige wenige konkurrierende Ontologien, auf die der Autor auch selbst eingeht. Aber sie erreichen bei weitem nicht den Umfang und die Komplexität derjenigen Ontologie, die der Autor im Rahmen seines Forschungsprojekts erstellt hat. Außerdem bestechen die Ausführungen des Autors dadurch, dass er alternative Gestaltungsoptionen für seine FMEA-Ontologie aufzeigt, die Vor- und Nachteile dieser Optionen diskutiert und sich schließlich für eine Alternative begründet entscheidet.

Schließlich präsentiert der Autor eine *prototypische Implementierung* eines Wissensbasierten Systems, das auf seiner Ontologie aufbaut und die Anwendung der FMEA-Technik in der betrieblichen Praxis unterstützen soll. Dieser Prototyp „OntoFMEA“ stellt ein webbasiertes, vornehmlich in Java implementiertes Softwaresystem dar, das vor allem für Interessenten aus dem Bereich der Wirtschaftsinformatik einige bemerkenswerte Anregungen für das Design von ontologiebasierten Wissensmanagementsystemen vermittelt. Das „Sahnehäubchen“ bildet die Integration der professionellen *Inferenzmaschine* Ontobroker. Sie gehört derzeit zu den weltweit führenden Inferenzmaschinen auf dem Gebiet von Ontologien und wurde von der deutschen Ontoprise GmbH, einem Spin-off der Universität Karlsruhe, entwickelt. Der Autor beweist eine glückliche Hand, seinen OntoFMEA-Prototyp so zu erläutern, dass auch „typisch betriebswirtschaftliche“ Leser, die weniger an informationstechnischen Details interessiert sind, gefesselt werden können. Die zahlreichen Screenshots vermitteln auf intuitive Weise einen guten ersten Überblick über die Funktionen, die vom Prototyp erfüllt werden.

Dem Autor ist es insgesamt auf vorzügliche Weise gelungen, die betriebswirtschaftliche FMEA-Technik für ontologiebasierte Wissensmanagementsysteme zu erschließen. Eine derart breit angelegte und zugleich tief greifende Analyse ist in diesem Bereich bislang noch nicht vorgelegt worden. Daher ist den vielschichtigen, erfrischend präzisen und oftmals mit überraschenden Details aufwartenden Ausführungen des Autors eine möglichst breite Resonanz unter allen betriebswirtschaftlich interessierten Lesern zu wünschen. Aber auch „Zaungäste“ u.a. aus den Bereichen der Wirtschaftsinformatik und der Angewandten Informatik sollten sich eingeladen fühlen, in diesem Werk bemerkenswerte Einblicke in die betriebswirtschaftliche Reflexion ihrer eigenen Arbeiten zu finden.

Stephan Zelewski

Vorwort

Das Management des Wissens eines Unternehmens wird in der Hauptsache durch zwei Problemstellungen erschwert oder sogar verhindert. Relevantes Wissen liegt erstens oftmals nicht in expliziter Form vor, wie zum Beispiel Datenbanken, sondern implizit in Dokumenten, wie zum Beispiel Projektberichten und Qualitätsmanagementhandbüchern. Darüber hinaus wird dieses Wissen in Dokumenten vorgehalten, die nur spezifischen Zwecken zugeordnet wurden. Dieses implizite Wissen ist nicht unmittelbar für "andere" Zwecke erreichbar, d. h. es kann bspw. nicht über ein herkömmliches Datenbanksystem akquiriert werden. Der Zugang zu Wissen wird zweitens durch das Problem erschwert, dass Akteure verschiedene Konzepte benutzen, um über dieselbe Thematik zu sprechen.

Gerade bei der Erfüllung operativer Aufgaben, die auf einer starken Arbeitsteilung basieren, können Ontologien (als Instrument des Wissensmanagements) helfen, relevantes Wissen zur Aufgabenerfüllung bereitzustellen, indem sie die Konzepte einer Domäne einheitlich strukturieren. Das Instrument Ontologien stammt aus dem Feld der Forschung zur Künstlichen Intelligenz. Die meistgenannte Definition hierzu geht zurück auf Gruber (1993): eine Ontologie ist „eine explizite Spezifikation einer Konzeptualisierung". Eine Ontologie wird in einem Wissensbasierten System genutzt, das die Möglichkeit zur Inferenz besitzt.

In den letzten Jahren ist die Fehlermöglichkeits- und Einflussanalyse (FMEA) als Instrument des Qualitätsmanagements in modernen Qualitätsmanagementsystemen hochgradig relevant geworden. Eine durchgeführte FMEA enthält Wissen über eine Domäne (z. B. zu bestimmten Produkten oder Prozessen), das genutzt werden kann, um Ontologien zu entwickeln. Diese Ontologien können genutzt werden, um ein IT-basiertes Wissensmanagementsystem aufzubauen. In der Umkehrung kann dieses ontologiebasierte Wissensmanagementsystem den Aufwand erleichtern, eine FMEA durchzuführen, weil das bereits existierende Wissen leicht wieder verwendet werden kann.

Ein Vorgehensmodell als ein weiteres Instrument des Wissensmanagements wird in der Arbeit vorgestellt, um zu zeigen, wie eine Ontologie aus FMEA-Formblättern zu erzeugen ist. Mit Hilfe dieses Vorgehensmodells wurde eine FMEA-Ontologie entwickelt. Diese FMEA-Ontologie wurde wiederum als Herzstück verwendet, um ein prototypisches Wissensbasiertes System (OntoFMEA-Prototyp) zu entwickeln. Mit seiner Ontologie ist OntoFMEA in der Lage, durch Wiederverwendung von Wissen aus früheren FMEAs, Vorschläge zu Systemelementen, Funktionen, Fehlern und Maßnahmen zu generieren.

Die vorliegende Dissertation entstand während meiner Zeit als wissenschaftlicher Mitarbeiter am Institut für Produktion und Industrielles Informationsmanagement der Universität Duisburg-Essen, Campus Essen. Die Entstehung des Werks wurde von vielen Personen unterstützt, bei denen ich mich bedanken möchte. In erster Linie bedanke ich mich bei meinem Doktorvater Herrn Univ.-Prof. Dr. Stephan Zelewski, der mir während der gesamten Zeit immer als Ansprechpartner hilfreich zur Seite stand. Die, dem offenen wissenschaftlichen Diskurs gewidmete, Atmosphäre an seinem Institut wird mir immer in dankbarer Erinnerung bleiben. Danken möchte ich auch Herrn Univ.-Prof. Dr. Stefan Eicker dafür, dass er sich bereit erklärte die Aufgaben des Korreferenten zu übernehmen, und Herrn Univ.-Prof. Dr. Christoph Lange als weiteres Mitglied meiner Prüfungskommission.

Wie jede wissenschaftliche Anstrengung verdankt auch diese Dissertation sich nicht allein den Bemühungen des Autors. Ich möchte hiermit allen Ratgebern für ihre Unterstützung danken. Insbesondere gebührt mein Dank Susanne Apke, Tim Rademacher und Joachim Penzel sowie meinen weiteren Ko-Autoren bei verschiedenen Aufsätzen, die während meiner Forschungstätigkeit entstanden sind.

Ganz herzlich möchte ich meiner Familie danken, die mich immer unterstützt hat und nicht zuletzt beim Korrekturlesen die Thematik durchdringen „musste“. Katinka Schneider widme ich dieses Buch.

Lars Dittmann

Inhaltsverzeichnis

Abkürzungs- und Akronymverzeichnis

A	Auftretenswahrscheinlichkeit
AAAI	American Association for Artificial Intelligence
ACL	Attributive Concept Descriptions Language
ACM	Association for Computing Machinery
AI	Artificial Intelligence / Künstliche Intelligenz
AIAI	Artificial Intelligence Applications Institute
AIFB	Angewandte Informatik und formale Beschreibungsverfahren
AG	Aktiengesellschaft
AL	Aussagenlogik
ANSI	American National Standards Institute
Appl/A	Ada Process Programming Language with Aspen
ARIS	Architektur integrierter Informationssysteme
Aufl.	Auflage
Ausg.	Ausgabe
B	Bedeutung
BSC	Balanced Scorecard
BTW	Business, Technologie und Web
bzw.	beziehungsweise
CAC	Codex Alimentarius Content
CAQ	Computer Aided Quality
CCP	Critical Control Point
CEUR	Central Europe
CG	Conceptual Graphs
CIKM	Conference on Information Knowledge Management
CIM	Computer Integrated Manufacturing
CLIPS	C Language Integrated Production System
CNC	Computerized Numerical Control
Cyc	Encyclopedia
CycL	Cyc Language (Wissensrepräsentationssprache)
DAML	DARPA Agent Markup Language
DARPA	Defense Advanced Research Projects Agency
DBW	Die Betriebswirtschaft
DGQ	Deutsche Gesellschaft für Qualität e.V.
d. h.	das heißt
DIAM	Defense Intelligence Agency Manual
DIN	Deutsches Institut für Normung
DL	Description Logics; Beschreibungslogik
DLR	Deutsches Zentrum für Luft- und Raumfahrt e.V.
DMT	Deutsche Montan Technologie GmbH
DoE	Design of Experiments / Statistische Versuchsplanung
DTD	Document Type Definition
E	Entdeckungswahrscheinlichkeit
ECAI	European Conference on Artificial Intelligence
EDV	elektronische Datenverarbeitung
EFFORT	Evaluation Framework for Ontologies and Related Technologies
EKAW	European Workshop on Knowledge Acquisition, Modeling and Management
EN	Europäische Norm
EPK	Ereignisgesteuerte Prozesskette
ER	Entity Relationship Ansatz
ERM	Entity-Relationship-Modell
ERP	Enterprise Resource Planning
ETA	Event Tree Analysis
et al.	et alius / et alii
EOQ	European Organization for Quality
e. V.	eingetragener Verein
F	Fehler
f.	folgende Seite
FAO	Food and Agriculture Organization of the United Nations
FIS	Fehler-Informations-System
FF	Fehlerfolge
ff.	folgende Seiten
FL	Frame Logic
F-Logic	Frame Logic
Florid	F-Logic Reasoning in Databases
FMEA	Fehlermöglichkeits- und Einflussanalyse, auch Failure Mode and Effects Analysis

FMECA	Failure Mode, Effects and Criticality Analysis
Fn.	Fußnote(n)
FOIS	Formal Ontologies in Information Systems
FOL	First Order Logics
FQS	Forschungsgemeinschaft Qualitätssicherung e. V.
FTA	Fault Tree Analysis
FU	Fehlerursache
GM	General Motors Company
GmbH	Gesellschaft mit beschränkter Haftung
HACCP	Hazard Analysis and Critical Control Point
HOL	Higher Order Logics
HRMS	Human Resource Management System
Hrsg.	Herausgeber
html	HyperText Markup Language
http	Hypertext Transfer Protocol
IAAI	Innovative Applications of Artificial Intelligence
ICADS	Intelligent Car Audio Design System
IEEE	Institute of Electrical and Electronics Engineers
IEC	International Electrotechnical Commission
i. e. S.	im engeren Sinn
ICCS	International Conference on Computational Science
IFIP	International Federation for Information Processing
IJCAI	International Joint Conference on Artificial Intelligence
Inc.	Incorporated
ISD	Intelligent Systems Division
ISI	Information Science Institute
ISMQC	International Symposium on Measurement and Quality Control in Production
ISO	International Organization for Standardization
IT	Informationstechnologie
i. w. S.	im weiteren Sinn
Jg.	Jahrgang
KACTUS	Modelling Knowledge about Complex Technical Systems for Multiple Use
KADS	Knowledge Analysis and Documentation System
Kap.	Kapitel
KAW	Knowledge Acquisition for Knowledge-Based Systems Workshop
KARL	Knowledge Acquisition and Representation Language
KBS	Knowledge-Based System
KBSI	Knowledge Based Systems Incorporated
K-CAP	Knowledge Capture
KE	Knowledge Engineering
KI	Künstliche Intelligenz
KIF	Knowledge Interchange Format
KOWIEN	Kooperatives Wissensmanagement in Engineering-Netzwerken
KSM	Karl Schumacher Maschinenbau GmbH
KM	Knowledge Machine (Repräsentationssprache)
KRDB	Knowledge Representation meets Databases Workshop
KSL	Knowledge Systems Laboratory
LISP	List Processing (Repräsentations-/Programmiersprache)
LNI	Lecture Notes in Informatics
M7	Seven New Tools / Sieben Qualitätswerkzeuge
MIKE	Modellbasiertes und Inkrementelles Knowledge Engineering
MIL-STD	Military Standard
MSL	Marvel Strategy Language
MVP-L	Multi View Process Modeling Language
NACMCF	National Advisory Committee on Microbiological Criteria for Foods
NASA	National Aeronautic and Space Agency
NAT	nicht-atomarer Term
NLP	Natural Language Processing
OCML	Operational Conceptual Modeling Language
ODE	Ontology Development Environment
OE	Ontology Engineering
o. Hrsg.	ohne Herausgeber
OIL	Ontology Inference Layer
OKBC	Open Knowledge Base Connectivity
OMG	Object Management Group
ONIONS	ONtologic Integration Of Naïve Sources
OntoFMEA	Ontologiebasierte FMEA
o. O.	ohne Ort
o. S.	ohne Seiten

OOP	Objektorientierte Programmierung
o. V.	ohne Verfasser
OWL	Web Ontology Language (auch vereinzelt: Ontology Web Language)
OWL DL	Web Ontology Language Description Logics
OXML	Ontology XML
PAKM	Practical Aspects of Knowledge Management
PDF	Portable Document File
PDM	Produktdatenmanagement
PIM	Produktion und Industrielles Informationsmanagement
Pkw	Personenkraftwagen
PL	Prädikatenlogik
PL1	Prädikatenlogik erster Stufe
PL2	Prädikatenlogik zweiter Stufe
PN	Petri-Netz
PROLOG	Programming in Logic
Q7	Seven QC Tools / Sieben Qualitätswerkzeuge /
QC	Quality Control / Qualitätslenkung
QFD	Quality Function Deployment / Qualitätsfunktionen-Darstellung
QM	Qualitätsmanagement
QS	Qualitätssicherung
QZ	Qualität und Zuverlässigkeit
RAMS	Reliability and Maintainability Symposium
RCP	Recommended Codes of Practice
RDF(S)	Resource Description Framework (Schema)
RPZ	Risikoprioritätszahl
RuleML	Rule Markup Language
S.	Seite(n)
SAE	Society of Automotive Engineers, Inc.
SCI	Multiconference on Systemics, Cybernetics, and Informatics
SE	Software Engineering
SE	Systemelement
SEU	Softwareentwicklungsumgebung
SGES	Specialist Group on Expert Systems
SGML	Standardized General Markup Language
SKSI	Semantic Knowledge Source Integration
s. o.	siehe oben
SPC	Statistical Process Control / Statistische Prozessregelung
St.	Sankt
s. u.	siehe unten
SWRL	Semantic Web Rule Language
TKDE	Transactions in Knowledge and Data Engineering
TOVE	Toronto Virtual Enterprise
TQM	Total Quality Management
u. a.	und andere / unter anderem
UML	Unified Modeling Language
URL	Unified Resource Locator
usw.	und so weiter
vs.	versus
VDA	Verband der Automobilindustrie e.V.
VDE	Verband Deutscher Elektrotechniker e.V.
VDI	Verband Deutscher Ingenieure e.V.
V-Modell	Vorgehens-Modell
W3C	World Wide Web Consortium
WBS	Wissensbasiertes System
WebODE	Web Ontology Design Environment
WHO	World Health Organization
WiSt	Wirtschaftswissenschaftliches Studium
WS	Wintersemester
WM	Wissensmanagement
WWW	World Wide Web
XML	extensible Markup Language
XPS	Expertensysteme
z. B.	zum Beispiel
ZfB	Zeitschrift für Betriebswirtschaftslehre
zfo	Zeitschrift Führung + Organisation

Symbolverzeichnis

deskriptive Symbole

C	Menge der Konzepte
f_i^k	Funktionssymbol
F	Menge der prädikatenlogischen Formeln
F_{fix}	Menge der Regeln, die Eigenschaften reflektieren
F_{spec}	durch die spezifische Domäne restringierte Regeln
O	Tupel der Modellierungsprimitive einer Ontologie in einem Universum
P_i^k	Prädikatssymbol
R	Menge der Relationen
R^H	hierarchische terminologische Relationen zwischen Konzepten
R^R	nicht-hierarchische terminologische Relationen zwischen Konzepten
t_k	Term
U	Universum
$\langle ... \rangle$	hervorgehobene Notation für Tupel

logisch-mathematische Symbole

$\in$	ist Element von
$\times$	Kreuzprodukt
$\{...\}$	Menge
$X \subseteq Y$	X ist Teilmenge von Y
$X \cup Y$	X vereinigt mit Y
$X \cap Y$	X geschnitten mit Y
$\varnothing$	leere Menge
$\vee$	Adjugat
$\wedge$	Konjugat
$\rightarrow$	Subjunktion
$\Rightarrow$	Implikation
$\neg$	Negat
$\leftrightarrow$	Bijunktion
$\Leftrightarrow$	Äquivalenz
$\exists$	Existenzquantor
$\forall$	Allquantor
*E*R.A	universelle Quantifikation
*A*R.A	existentielle Quantifikation
+	Addition
-	Subtraktion
:	Division
*	Multiplikation
=	ist gleich
<	kleiner als
>	größer als

Abbildungs- und Tabellenverzeichnis

Abbildungen

Tabellen

1 Einführung

1.1 Wirtschaftswissenschaftliche Problemeingrenzung

1.1.1 Wettbewerbliches versus instrumentelles Umfeld

Grundsätzlich lässt sich die auf den nächsten Seiten vorgestellte Problemstellung der Arbeit dem Gebiet der Instrumente[1] für die betriebliche Leistungserstellung zuordnen. Instrumente werden genutzt, um einen Produktionsprozess hinsichtlich der eingesetzten Ressourcen zu gestalten. Besonderes Augenmerk lässt sich auf die integrierte Verwendung unterschiedlicher Instrumente für eine gemeinsame Zielerreichung innerhalb eines Unternehmens als instrumentelles Umfeld legen. Darüber hinaus befinden sich Unternehmen mit ihrer Leistungserstellung immer auch in einem wettbewerblichem Umfeld, das als übergeordnete „Instanz“ Unternehmen einheitliche Rahmenbedingungen vorgibt.

Das instrumentelle Umfeld wirkt auf das wettbewerbliche Umfeld eines Unternehmens, indem es Möglichkeiten zur Verfügung stellt, die beispielsweise eine Umsetzung von Wettbewerbsstrategien, die dem wettbewerblichen Umfeld zugeordnet werden, überhaupt erst ermöglichen. Innerhalb des wettbewerblichen Umfelds ergibt sich für die vorliegende Arbeit eine Problemstellung aus den in der Literatur zu findenden Wettbewerbsstrategien. Aus dem Grund der konstituierenden Abhängigkeit von instrumentellem und wettbewerblichem Umfeld wird zunächst tiefer auf das wettbewerbsstrategische Problemfeld eingegangen, um den Gesamtrahmen der Arbeit abzustecken, und anschließend wird sich dem instrumentellen Problemfeld gewidmet, dem sich die weiteren Ausführungen der Arbeit unmittelbar zuordnen lassen.[2]

Zusammenfassend lässt sich feststellen: Die vorliegende Arbeit soll mittelbar im wettbewerblichen Umfeld wirken, indem sie hauptsächlich im instrumentellen Umfeld eine Anwendung vorstellt, die Instrumente des Wissens- und des Qualitätsmanagements integriert und sich dadurch bei erfolgreichem Einsatz auf die wettbewerbliche Situation eines Unternehmens positiv auswirkt. Dabei soll die Arbeit im Wesentlichen die These der Möglichkeit von hybriden Wettbewerbsstrategien unterstützen.

---

1) Der Begriff „Instrument“ umfasst im hier verwendeten Sinn alle Hilfsmittel, die während einer Tätigkeit zur Erreichung eines Ziels eingesetzt werden können, um dieses Ziel effizienter oder effektiver zu erreichen, als es ohne den Einsatz des Hilfsmittels der Fall sein würde. Darüber hinaus können Instrumente notwendig sein, um überhaupt das Ziel einer Tätigkeit erreichen zu können. Instrumente können sowohl gegenständlich als auch konzeptuell vorliegen. Zahlreiche Beispiele verschiedener Instrumente finden sich in den Ausführungen der Kapitel 3, 4 und 5 ab Seite 35 ff.

2) Als Kontext zur wissenschaftlichen Problemstellung im instrumentellen Umfeld wird ein wettbewerbliches Umfeld benötigt, es steht jedoch nicht im Fokus des wissenschaftlichen Erkenntnisinteresses dieser Arbeit. Deshalb lassen sich die vorliegenden wettbewerbsstrategischen Ausführungen eher als „Nebenproblem“ einordnen, die den gesetzten Rahmen der weiteren Ausführungen wiedergeben.

1.1.2 Wettbewerbliches Umfeld

1.1.2.1 Wettbewerbsstrategien gemäß Porter

Die aktuelle Wettbewerbssituation produzierender Unternehmen ist auf zahlreichen Absatzmärkten durch Kostenkonkurrenz und Qualitätsdruck (bspw. in der Automobil- und Bauindustrie) gekennzeichnet.[3] Zusätzlich führt Adam an, dass der für viele Märkte geltende Wandel (hin zum Käufermarkt) zu einer nachhaltig gesteigerten Komplexität in den Unternehmen geführt hat.[4] Folgt man der hier unterstellten Annahme, dass die Beherrschung dieser Komplexität durch Wissen[5] sichergestellt werden kann und der Einsatz von Wissen vornehmlich der Vermeidung von Fehlern[6] bei der Leistungserstellung dient, so lässt sich konstatieren: Unternehmen befinden sich zunehmend in einem Wettbewerb, der neben der Kostenkontrolle unter anderem eine fortwährende Reduzierung von Fehlern im Herstellungsprozess fordert.[7]

Gemäß Porter lassen sich drei generische Strategien für Unternehmen als mögliche Antwort auf die skizzierte Wettbewerbssituation zur erfolgreichen Marktbearbeitung unterscheiden:[8]

1. umfassende Kostenführerschaft,
2. Differenzierung und
3. Konzentration auf Schwerpunkte.

Grundsätzlich kann das Ziel eines dauerhaften Wettbewerbsvorteils gemäß Porter entweder durch Kostenführerschaft, d. h. Minimierung der realen Stückkosten bei lediglich angemessener Qualität der Leistung, oder durch Differenzierung, d. h. Anbieten einer Leistung, die vom

---

3) Vgl. zum Marktwandel Adam (1998), S. 27 ff.; Müller (1991), S. 781 ff. Vgl. zum Wettbewerb im Wandel: Farmer, Vlk (2005), S. 23 ff.; Jost (2005), S. 219 f.; Ludwig (1998), S. 19 ff.; Thiele (1997), S. 1 ff. Unternehmen sehen sich einem steigenden Qualitätsdruck (im Sinne eines Qualitätswettbewerbs) ausgesetzt, der bspw. auf die Internationalisierung der Märkte zurückzuführen ist (vgl. Kamiske, Malorny (1994), S. 3).
Um die Verständlichkeit der Arbeit zu erleichtern, wird im Folgenden einheitlich mit Beispielen aus der Automobil- und Maschinenbauindustrie gearbeitet.

4) Vgl. Adam (1998), S. 30. Dabei bezieht sich die zunehmende Komplexität in den Unternehmen insbesondere auf die Phase der Produktentwicklung. Es wird in interdisziplinären, unternehmensübergreifenden Projektteams gearbeitet (vgl. Corsten (1998), S. 35).

5) Siehe zum Begriff „Wissen“ Kapitel 2.1.2.1, S. 22 f. Zunehmend wird Wissen und der Umgang damit als wettbewerbsentscheidende Einflussgröße angesehen, um die Komplexität zu reduzieren (vgl. North (2002), S. 9 ff.). Entscheidend ist dabei, dass die Ressource Wissen zur richtigen Zeit am richtigen Ort in der richtigen Menge verfügbar wird, um den größtmöglichen Nutzen für das Unternehmen zu generieren.

6) Siehe zum Begriff „Fehler“ Kapitel 2.1.1.3, S. 20 f.

7) Vgl. Müller (1991), S. 783; Pfeifer (2002), S. XXIII f.

8) Vgl. Porter (1999), S. 70 ff.

Kunden als einmalig wahrgenommen wird (bspw. als „hohe“ Qualität), erreicht werden.[9] Als dritte Möglichkeit bleibt gemäß Porter die Konzentration auf Schwerpunkte, d. h. es werden Marktnischen besetzt.[10]

1.1.2.2 Qualität als Einflussgröße

In jedem Fall der drei Wettbewerbsstrategien von Porter wird Qualität als Einflussgröße auf den Erfolg (z. B. Gewinn oder Return on Investment) anerkannt.[11] Die Erfolgswirkung der Einflussgröße Qualität wurde u. a. von Buzzell und Gale untersucht. Die Autoren identifizierten Gründe für die Kosten- und Erlöswirkung von Qualität durch ihr PIMS-Programm[12] (siehe hierzu Abbildung 1).[13] Zu unterscheiden ist dabei nach Buzzell und Gale zunächst zwischen technischer und wahrgenommener Qualität.[14] Bei der technischen Qualität handelt es sich um die Erfüllung technischer Standards und bei der wahrgenommenen Qualität handelt es sich um die subjektiven Eindrücke der Kunden. Die Strategie der Qualitätsführerschaft zielt in erster Linie auf die wahrgenommene Qualität ab. Die Strategie der Kostenführerschaft zielt auf

---

9) Fleck stellt hierzu ein Klassifikationsschema von Differenzierungsstrategien, das eine Innovations-, eine Varietäts- und eine Qualitätsstrategie berücksichtigt, vor (vgl. Fleck (1995), S. 88). Im Folgenden konzentrieren sich die Ausführungen dieser Arbeit bei der Differenzierungsstrategie auf den Typ der Qualitätsführerschaft (Qualitätsstrategie) aus drei Gründen. Erstens wird im Hauptteil der Arbeit ein Instrument des Qualitätsmanagements eingesetzt. Zweitens wird selbst von Porter Qualität immer in einer Mindestausprägung vorausgesetzt, d. h. bei der Produktion kann der Aspekt Qualität niemals vollständig ausgeblendet werden (siehe hierzu auch die Ausführungen im folgenden Kapitel). Drittens lassen sich bei ausreichend breiter Auffassung des Konstrukts „Qualität“ sämtliche Unterscheidungen Porters für Formen der Differenzierung hierunter einordnen (vgl. hierzu bspw. Luchs, Neubauer (1986), S. 9).

10) Bewerkenswert bei den gemachten Aussagen ist, dass bei Porter (und auch bei Corsten) der Faktor Wissen nicht explizit Verwendung findet. Vielmehr wird Wissen implizit den Strategien zugeordnet, indem stillschweigend davon ausgegangen wird, dass das notwendige Know-how für eine *Kostenführerschaft*, d. h. für die Produktion zu niedrigsten Stückkosten, vorhanden sein muss oder sich aber akquirieren lässt (Gleiches gilt auch bei Verfolgung der anderen Strategien: *Differenzierung* und *Konzentration auf Schwerpunkte*).

11) Porter geht dabei von mindestens “angemessener” Qualität aus, d. h. selbst bei der Kostenführerschaft vertritt Porter die Meinung, dass bei Nichteinhaltung von Mindestqualitätsstandards kein dauerhafter Wettbewerbsvorteil gesichert werden kann (vgl. Porter (1999), S. 71). Auch bei der Konzentration auf Schwerpunkte lässt sich Qualität als Einflussgröße auf den Erfolg festmachen, denn gemäß Porter führt dieser Strategietyp im Ergebnis zu einer Differenzierung oder niedrigen Kosten (vgl. Porter (1999), S. 75).

12) Vgl. Buzzell, Gale (1989). Eine kurze Übersicht zum PIMS-Programm findet sich in Jacob (1983). Zu einer Kritik am PIMS-Programm siehe z. B. Jacob (1983), S. 265, und Kreikebaum (1997), S. 116.

13) Vgl. Gale, Buzzell (1989), S. 7. Vgl. hierzu auch Luchs, Neubauer (1986), S. 22, und Zenz (1999), S. 155. Die Untersuchung des ROI (Return on Investment) innerhalb des PIMS-Programms in Abhängigkeit von der Produktqualität zeigt, dass 20% aller untersuchten Unternehmen, die eine unterlegene relative Produktqualität erzielten, einen ROI von ca. 16% aufwiesen. Demgegenüber konnten die 20% der Unternehmen mit einer überlegenen relativen Produktqualität einen ca. doppelt so hohen ROI realisieren (vgl. Buzzell, Gale (1989), S. 93). Zum Begriff „relativ“ siehe Fn. 15.

14) Vgl. Buzzell, Gale (1989), S. 90. Insgesamt lassen sich fünf Qualitätsbegriffe in der Literatur differenzieren. Siehe hierzu Kapitel 2.1.1.1, S. 16 f. Die technische Qualität fällt dabei unter den produktbezogenen Qualitätsbegriff und die wahrgenommene Qualität unter den kundenbezogenen Qualitätsbegriff.

die relative[15] Effizienz bei der Erfüllung von Spezifikationen (d. h. Mindeststandards müssen erfüllt werden) ab, d. h. die technische Qualität rückt in den Fokus der Unternehmensanstrengungen bei Durchsetzung einer Strategie der Kostenführerschaft.

Für eine vom Kunden wahrgenommene höhere relative Qualität (bei Verfolgung einer *Strategie der Qualitätsführerschaft*) kann ein höherer relativer Preis verlangt werden.[16] Ein höherer durchsetzbarer Preis steigert die Rentabilität des Unternehmens. Wird stattdessen der Preis nicht erhöht, so verbessert sich aufgrund des gestiegenen Nutzens für den Kunden das relative Nutzen/Preis-Verhältnis des angebotenen Produkts. Dies führt zu einer Steigerung des relativen Marktanteils, weil vermehrt Kunden auf dieses Produkt zugreifen werden. Ein höherer relativer Marktanteil steigert zum einen direkt die Rentabilität eines Unternehmens (z. B. aufgrund steigender Marktmacht) und zum anderen indirekt über Skaleneffekte. Aufgrund größerer Stückzahlen und zunehmender Erfahrung bei deren Produktion können – durch diese so genannten Skaleneffekte – die relativen Kosten gesenkt werden. Dies führt wiederum zu einer erhöhten Rentabilität des Unternehmens.

Bei Verfolgung der *Strategie der Kostenführerschaft* müssen die vom Markt mindestens geforderten Spezifikationen für ein Produkt bei der Herstellung erfüllt werden. Dies lässt sich gemäß Buzzell und Gale durch eine überlegene technische Qualität innerhalb des Produktionsprozesses erreichen.[17] Die Umsetzung überlegener technischer Qualität bei Verfolgung einer Strategie der Kostenführerschaft bringt zwei Vorteile mit sich. Zum einen sind die (Gesamt-)*Kosten der Qualität* niedriger als bei Durchsetzung einer Qualitätsführerschaft und zum

15) Der Begriff „relativ" steht hier als Platzhalter für die Bezeichnung, dass eine Betrachtungseinheit eines Unternehmens in Bezug zu setzen ist mit einer adäquaten Betrachtungseinheit der Konkurrenz (i. d. R. wird der „stärkste" Konkurrent als adäquate Betrachtungseinheit angesetzt). So bedeutet bspw. „relative Effizienz", dass ein betrachtetes Unternehmen ein Produkt bei gleich bleibender Spezifikationserfüllung relativ zu seinen Konkurrenzunternehmen effizienter (mit geringeren Kosten) herstellen kann.

16) Diese Aussage wird bspw. auch durch die Untersuchung der Deutschen Gesellschaft für Qualität e. V. gestützt, die ermittelte, dass die Aussage „Gute Qualität lasse ich mir auch gerne etwas kosten" mit durchschnittlich 77 von maximal 100 Zustimmungspunkten bewertet wurde (ermittelt aus 1338 Interviews mit einer bevölkerungsrelevanten Stichprobe Deutschlands; vgl. ExBa (2003), S. 19). Dieser Wert stellt zugleich die höchste Ausprägung aller Aussagen, die bei der Untersuchung der Motive für das Kaufverhalten berücksichtigt wurden, dar. So erhielt bspw. die Aussage „Ich finde es sehr gut, wenn der Service zurückgeht und dafür die Preise sinken." lediglich einen Mittelwert von 47,9 Zustimmungspunkten.

17) An dieser Stelle sei abschließend auf die unterschiedlichen Dimensionen der Begriffsverwendung von „Qualität" hingewiesen. Eine Wettbewerbsstrategie der Differenzierung in Form eines Qualitätswettbewerbs gemäß Porter führt zu einer verbesserten Ausprägung der durch den Kunden wahrgenommenen Qualität. Während eine überlegene technische Qualität im Sinne von Buzzell und Gale sich auf den Produktionsprozess erstreckt und sich bspw. in einer geringen Ausschussquote widerspiegelt. Diese überlegene technische Qualität kann durchaus zu Produkten führen, die vom Kunden als qualitativ geringwertig wahrgenommen werden.

anderen wird die erfüllte technische Qualität vom Kunden wahrgenommen.[18] Der erste Vorteil führt zu niedrigeren Gesamtkosten und damit zu einer Rentabilitätssteigerung, weil im Gegensatz zur Verfolgung einer Qualitätsführerschaft bspw. externe Kosten nur in geringerem Maße anfallen, in dem auf eine weitreichende Kulanz gegenüber den Kunden verzichtet wird und lediglich die gesetzlich vorgeschriebene Gewährleistung eingehalten wird. Der zweite Vorteil führt ebenfalls zur Akzeptanzsteigerung hinsichtlich eines relativ höheren durchsetzbaren Preises beim Kunden, wenn auch nicht in dem Maße, wie es bei einer Strategie der Qualitätsführerschaft der Fall sein würde, denn die Verbesserung der vom Kunden wahrgenommenen Qualität bezieht sich nur auf den unmittelbaren Bereich der Erstellung eines Produkts.[19]

Die Abbildung 1 fasst den skizzierten Sachverhalt zusammen.

---

18) Vgl. zu *Qualität und Kosten*: Bruhn, Georgi (1999); Feigenbaum (1991), S. 109 ff.; Pfeifer (2002), S. 181 ff.
Im Bereich des Qualitätsmanagements findet sich häufig eine Gliederung von *Qualitätskosten* (auch Kosten der Qualität) in drei Arten: *Fehlerverhütungskosten* (prevention costs), *Prüfkosten* (appraisal costs) und *Fehlerfolgekosten* (failure costs) (Feigenbaum (1991), S. 111). In dieser weit verbreiteten Kostengliederung, die auch als PAF-Schema bekannt ist, gehören zu den Fehlerverhütungskosten die Kosten der Qualitätsplanung und -lenkung (z. B. Kosten für Schulungsmaßnahmen). Sie können auch als Vorbeugungskosten verstanden werden und sind spezifisch zur Vermeidung einer nicht anforderungsgerechten Qualität. Die Prüfkosten ergeben sich bei der Durchführung von Qualitätsprüfungen (z. B. Produktausgangsprüfung bei einem Fertigteilwerk). Bei den Fehlerfolgekosten wird zwischen internen und externen Kosten unterschieden. Interne Kosten fallen bei Beseitigung von Fehlern an, bevor der Kunde in den Erstellungsprozess integriert wird. Externe Kosten hingegen fallen z. B. bei einer Nachtragsforderung oder einem Gewährleistungsanspruch an, die ebenfalls aufgrund eines fehlerhaften Produkts entstehen.

19) Dieser Vorteil äußert sich bspw. in dem Umstand, dass ein Produkt immer derselben wahrgenommenen Qualität entspricht und es keinerlei Abweichungen innerhalb einer Charge gibt. Für den Kunden ergibt sich hieraus möglicherweise ein Argument für den Kauf des Produkts aufgrund der Tatsache, dass der Kunde immer dieselbe wahrgenommene Qualität erhält, wie bei vorangegangen Käufen des Produkts. An dieser Stelle wird dem einfachen Sachverhalt aus Fn. 15 gefolgt und unterstellt, dass auch eine technisch überlegene Qualität zu einem höheren durchsetzbaren Preis führt.

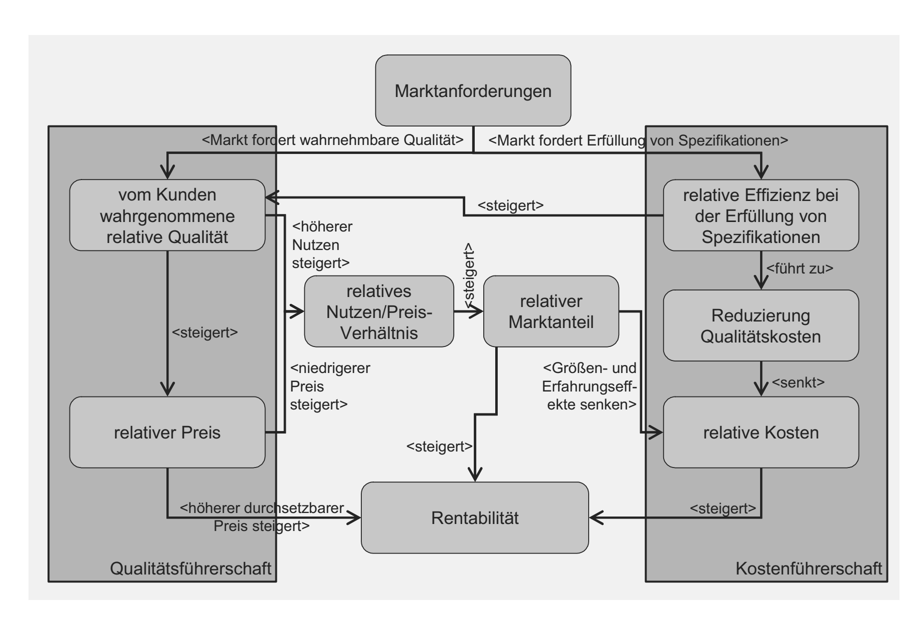


Abbildung 1: Erfolgswirkung der Qualität[20]

20) Vgl. Gale, Buzzell (1989), S. 92. Vgl. aber auch Luchs, Neubauer (1986), S. 22, und Zenz (1999), S. 155.

1.1.2.3 Hybride Wettbewerbsstrategien

Im Ansatz von Buzzell und Gale findet sich ein erster Anhaltspunkt für die Umsetzung einer hybriden Strategie, die sowohl eine Kostenführerschaft als auch eine Qualitätsführerschaft (im Sinne der Umsetzung einer Wettbewerbsstrategie der Differenzierung) verfolgt, in dem zwischen technischer und wahrgenommener Qualität unterschieden wird, die beide als Qualität auf den Erfolg eines Unternehmens zeitgleich wirken können.

In der Literatur finden sich hierzu weitere Ansätze, die die Integration der generischen Strategien Porters zu einer hybriden Wettbewerbsstrategie propagieren.[21] Dabei widersprechen sie Porters Überlegung von der prinzipiellen Unvereinbarkeit der genannten generischen Strategien.[22] In diesem Zusammenhang weisen Corsten und Will auf die Bedeutung von informationstechnisch basierten Systemen hin, weil mit ihrer Hilfe simultan Produktivität, Flexibilität und Qualität positiv beeinflusst werden können.[23]

Die Anwendung von informationstechnisch basierten Systemen und deren Instrumente fallen unter den Begriff „Wissensmanagement“.[24] Anerkannt wird dabei, dass Wissensmanagement[25] und Qualitätsmanagement[26] die gleichen Ziele verfolgen.[27] Zu diesen Zielen gehören bspw. die Explizierung implizit vorhandenen Wissens für die Produktentwicklung und die gemeinschaftliche Lösung betrieblicher Problemstellungen.[28] Die Verbindung von Instrumenten des Wissens- und des Qualitätsmanagements bietet Erfolg versprechende Mög-

21) Vgl. Corsten (1998), S. 110 ff.; Fleck (1995); Welge, Al-Laham (2003), S. 395 ff. Zu einer ersten Unterscheidung von hybriden Wettbewerbsstrategien siehe Proff (1997), S. 306. Auf die Unterscheidung hinsichtlich Sukzession (vgl. Gilbert, Strebel (1987), S. 28) und Simultanität kann an dieser Stelle verzichtet werden, weil der Verfasser der Ansicht ist, wenn der Nachweis einer simultanen hybriden Wettbewerbsstrategie gelingt, es sich so darstellt, dass dies als hinreichend für eine sukzessive hybride Wettbewerbsstrategie anzusehen wäre.

22) Siehe hierzu die Ausführungen Porters zu Unternehmen, die sich “zwischen den Stühlen sitzend” befinden, sofern nicht einer der drei Strategietypen verfolgt wird. Anschaulich wird dies von Porter auch als U-Kurve von Rentabilität und Marktanteil dargestellt (vgl. Porter (1999), S. 78 ff.).

23) Vgl. Corsten (1998), S. 127 ff.; Corsten, Will (1994), S. 262 ff. Die „Produktivität“ bei Corsten und Will bezieht sich auf die oben genannte technische Qualität und die „Qualität“ von Corsten und Will bezieht sich auf die Dimension der oben genannten wahrgenommenen Qualität. Die Flexibilität wird an dieser Stelle vernachlässigt, weil sie für die Anschlussfähigkeit der Argumentation von nachrangiger Bedeutung erscheint. Zudem wird die andere von Corsten und Will genannte Möglichkeit für eine hybride Wettbewerbsstrategie der Gruppenarbeit in dieser Arbeit vernachlässigt, weil der Fokus der Arbeit auf dem Informationsmanagement liegt.

24) Vgl. Maier (2002), S. 25 f.

25) Vgl. North (2002); Probst, Raub et al. (2003); Nonaka, Takeuchi (1995).

26) Siehe hierzu die Ausführungen in Kapitel 2.1.1, S. 16 ff.

27) Vgl. Keller, Kuhn (2004), S. 12; Pfeifer (2001), S. 143; Woll, Zehl et al. (2001), S. 345 ff.

28) In diesem Zusammenhang identifiziert Johannsen vier Basisprozesse des Wissensmanagements (Erzeugung, Akkumulation, Austausch und Verwendung von Wissen), die mit der Hilfe von Instrumenten des Qualitätsmanagements erfolgreich bearbeitet werden können (vgl. Johannsen (2000), S. 45 ff.). Weitere Ausführungen zu Wissens- und Qualitätsmanagement finden sich in Kapitel 2, S. 16 ff.

lichkeiten zur Steigerung des Unternehmenserfolgs.[29] Die Verbindung kann insbesondere den Qualitätsdruck, d. h. die vom Kunden geforderte (wahrgenommene) Qualität erreichen zu müssen, kanalisieren und durch erfolgreiche Anwendung auch der Kostenkonkurrenz entgegentreten. Durch eine Kundenorientierung, die ein umfassendes Qualitätsverständnis beinhaltet, können Konkurrenzvorteile erlangt werden.[30]

Dem Verfasser bekannte Ansätze zur Verbindung von Wissens- und Qualitätsmanagement verbleiben auf der Meta-Ebene, d. h. sie gehen nicht weiter auf die *Verbindung von Instrumenten des Qualitäts- und des Wissensmanagements* zur erfolgreichen Umsetzung eines integrierten Ansatzes, der eine hybride Wettbewerbsstrategie ermöglicht, ein.[31] Hier eröffnet sich eine Diskrepanz, die sich in einer mangelhaften Umsetzung und damit Nachvollziehbarkeit der vorgeschlagenen Integration von Wissens- und Qualitätsmanagement äußert: Die in der Literatur vorhandenen Ansätze bleiben insofern unkonkret, als dass sie dem Rezipienten eine spezifische Anwendung anhand konkreter Instrumente vorenthalten.[32] Die Problemstellung der mangelhaften Nutzbarkeit der vorgeschlagenenen Managementansätze für Strategien muss eine Konkretisierung durch anwendbare Umsetzungsmöglichkeiten erfahren. Diese Umsetzung bezieht sich immer auf eine instrumentelle Umsetzung.

1.1.3 Instrumentelles Umfeld

1.1.3.1 Instrumentelle Konkretisierung des wettbewerblichen Umfelds

Das bisher in der Literatur zu Findende über hybride Wettbewerbsstrategien und deren verwendete Instrumente bleibt abstrakt und an der Oberfläche, so dass eine Implementierung kaum möglich wird.[33] Es bedarf deshalb einer instrumentellen Konkretisierung in Form einer

---

29) Vgl. hierzu die Aufsätze von Pfeifer, Scheermesser et al. (2000) und Kalbfleisch, Schellenberg et al. (2001). Es existieren einige wenige wirtschaftswissenschaftliche Arbeiten mit Beiträgen zu Ansätzen zur Steigerung der Wettbewerbsfähigkeit von Unternehmen mit dem Fokus auf dem Umgang mit *Wissen* und *Qualität.* Sie gehen nicht auf instrumenteller Ebene auf die Verbindung ein. Für Ansätze, die Wissensmanagement mit Qualitätsmanagement verbinden, vgl. Bickenbach, Freyler at al. (2000); Johannsen (2000); Kalbfleisch, Schellenberg et al. (2001); Pfeifer, Scheermesser et al. (2000); Woll, Zehl et al. (2001); Woll, Zehl (2001).

30) Vgl. Adam (1998), S. 29. Für Adam bedeutet dies, dass die vom Kunden wahrgenommene Qualität sich nach dem Grad der Befriedigung der kaufrelevanten Dimensionen des Kunden misst.

31) Corsten und Will weisen darauf hin, dass die „Suche nach Produktionskonzepten zur simultanen Unterstützung strategischer Erfolgsfaktoren an einer Integration neuerer informationstechnologischer und arbeitsorganisatorischer Gestaltungskonzepte ansetzen“ soll (Corsten, Will (1994), S. 272). Um jedoch diese beiden Konzepte integrieren zu können, ist es zunächst notwendig, die einzelnen Instrumente, die den jeweiligen Konzepten zu Grunde liegen, zu integrieren, weil erst eine Integration untereinander und anschließend eine Integration über die spezifischen Einsatzgebiete hinaus plausibel Erfolg versprechend erscheint.
Im vorliegenden Fall lässt sich die Ebene der Verbindung von Instrumenten als *Objekt-Ebene* zur Umsetzung einer hybriden Wettbewerbsstrategie begreifen.

32) In diesem Sinne folgt der Verfasser der Auffassung, dass eine „echte“ Verbindung von Wissens- und Qualitätsmanagement immer auch eine Verbindung der Instrumente, die zur Umsetzung der jeweiligen Managementstrategie zwingend benötigt werden, darstellt, damit eine einheitliche (hybride) Strategieumsetzung erfolgen kann.

33) So bspw. bei Corsten, Will (1994), S. 264 ff.; Proff (1997), S. 305 ff.; Welge, Al-Laham (2003), S. 395 ff.

Verbindung von Instrumenten des Wissens- und Qualitätsmanagements, die in dieser Arbeit angestrebt wird.[34]

Einzelne Instrumente sind zumeist unvollständig in Bezug auf ihre Abdeckung des gesamten betriebswirtschaftlichen Leistungsprozesses und in der Anwendung bisher nur auf der Objekt-Ebene der Produktionsprozesse berücksichtigt. Eine Untersuchung ihrer Eignung in Bezug auf die Unterstützung einer hybriden Wettbewertbsstrategie findet nicht statt.[35]

Demgegenüber steht, dass Teile der wirtschaftswissenschaftlichen Forschung eine Vollständigkeit der Analyse und Integration wirtschaftswissenschaftlicher Instrumente hinsichtlich ihres Beitrags zur Realisierung von (hybriden) Wettbewerbsstrategien anstreben.[36] Hierzu gehört auch eine Betrachtung der Auswirkungen der Integration der Instrumente-Anwendung der Objekt-Ebene auf die Meta-Ebene von Wissens- und Qualitätsmanagement und deren Auswirkung auf eine hybride Wettbewerbsstrategie.[37] Das Wissen über die Verwendung von Instrumenten für Produktionsprozesse auf der Objekt-Ebene soll expliziert, computergestützt verarbeitbar und dauerhaft wieder verwendbar gemacht werden, um im Wettbewerb bestehen zu können.[38] Die Abbildung 2 fasst die Situation zusammen. Es ergibt sich die folgende erläuterte Problemstellung der Arbeit.

---

34) Als beispielhafte Instrumente mit einer Eignung für hybride Wettbewerbsstrategien lassen sich gemäß Corsten Gruppenarbeit und Computer Integrated Manufacturing (CIM) nennen (vgl. Corsten (1998), S. 127 ff.).

35) Bis heute existiert in der Praxis kein Anwendungssystem, das als ein vollständiges Computer Integrated Manufacturing System (CIM-System) bezeichnet werden dürfte. Alle bisherigen Ansätze bleiben unvollständig und sind als Insellösungen anzusehen.

36) Bamberger, Wrona (2004), S. 388 ff.; Jost (2005), S. 234; Welge, Al-Laham (2003), S. 394 ff. Vgl. zu einem Ansatz, der Qualitätsmanagement mit Informationsmanagement verbindet, Hennig (2001); zu einem Ansatz, der Qualitätsmanagement mit Projektmanagement verbindet, Walder, Patzak (1997); zu einem Ansatz, der die Konzentration auf Kernkompetenzen mit der Balanced Scorecard verbindet, Homp, Danner (2002); zu Ansätzen, die Qualitätsmanagement mit Wissensmanagement verbinden, siehe Fußnote Fn. 29.

37) Dies geschieht etwa bei Bickenbach, Freyler at al. (2000), S. 351 ff.; Johannsen (2000), S. 42 ff.; Keller, Kuhn (2004), S. 12 ff.; Woll, Zehl et al. (2001), S. 345 ff. und den Autoren die unterschiedliche Ansätze verbinden in der voranstehenden Fußnote.

38) Vgl. Ullman (2002), S. 63.

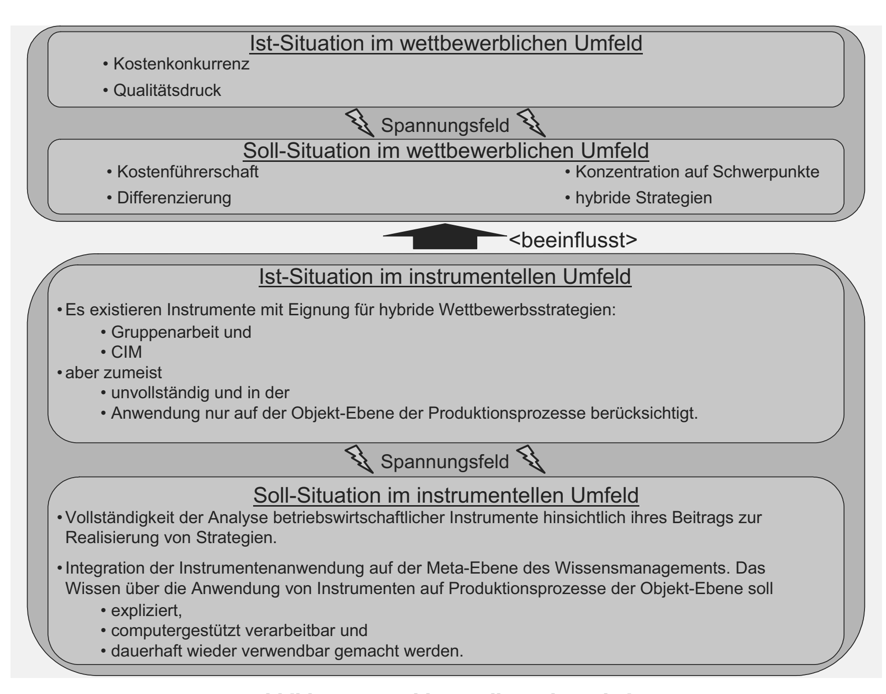


Abbildung 2: Problemstellung der Arbeit

Aus dem instrumentellen Umfeld eröffnet sich eine forschungsrelevante Diskrepanz bei der Umsetzung einer hybriden Wettbewerbsstrategie bezüglich einer Ist-Situation, die geprägt ist von der theoretisierten Möglichkeit der Existenz geeigneter Instrumente für die Umsetzung hybrider Wettbewerbsstrategien. Diese sind in ihrer Umsetzung bisher unvollständig und in der konkreten Anwendung nur auf die Objekt-Ebene der Produktionsprozesse beschränkt. Die Diskrepanz ergibt sich aus einer der Ist-Situation gegenüber stehenden Soll-Situation, die eine tatsächliche Explikation und damit eine dauerhafte computergestützte Wiederverwendung von Wissen über die Anwendung von integrierten Instrumenten auf Produktionsprozesse der Objekt-Ebene wünscht. Die *wissenschaftliche Problemstellung* der vorliegenden Arbeit ergibt sich aus der gewünschten Entwicklung eines integrierten Ansatzes, der betriebswirtschaftliche Instrumente in einer Weise zusammenführt, dass eine hybride Wettbewertbsstrategie, die zeitgleich eine Qualitäts- und Kostenführerschaft anstrebt, erfolgreich umgesetzt werden kann. Es wird eine instrumentelle Konkretisierung des wettbewerblichen Umfelds angestrebt.

1.1.3.2 FMEA, Ontologien und Vorgehensmodelle als Instrumente

Wie bereits erwähnt, führt das wettbewerbliche Umfeld zu Kostenkonkurrenz und Qualitätsdruck. Als eine Möglichkeit, diesem Umstand simultan zu begegnen, bietet sich die Fehlerreduzierung bei der Produktion an. Um eine Reduzierung zu erreichen, wird eine Fehleranalyse notwendig. Als Folge der Fehleranalyse können Fehler vermieden und Gesamtkosten durch die Reduzierung von Qualitätskosten (insbesondere Fehlerfolgekosten) gesenkt werden. Im

instrumentellen Umfeld dient insbesondere die *FMEA* (Fehlermöglichkeits- und Einflussanalyse) als ein Instrument zur präventiven Fehleranalyse. Sie findet seit geraumer Zeit weit reichende Anwendung[39] in der industriellen Entwicklung und Produktion und gilt als Stand der Technik.[40]

Eine systematisch durchgeführte FMEA strukturiert auf semi-formaler Ebene Wissen über Fehlerursachen und -wirkungen für einen spezifischen Anwendungshintergrund (Domäne), insbesondere dann, wenn sie IT-basiert durchgeführt wird.[41] Das Potential der Fehleranalyse wird jedoch nicht ausreichend genutzt, weil das inhärente Wissen oftmals nicht für eine Wiederverwendung aufbereitet wird:[42] Das mittels der Fehleranalyse bereits explizierte Wissen soll mit dieser Arbeit, im Gegensatz zur derzeit oftmals üblichen Vorgehensweise, die lediglich eine einmalige Verwendung bedeutet[43], einer dauerhaften Verwendung auch in weiteren Instrumenten zugeführt werden, um den Unternehmensfortbestand nachhaltig zu sichern.[44] Durch die erfolgreiche Wiederverwendung von Wissen lassen sich die Kosten – aufgrund der Vermeidung von Doppelarbeiten – weiter verringern.

Grundsätzlich stellt Johannsen fest, dass Instrumente des Qualitätsmanagements genutzt werden können, um Wissen zu transferieren; allerdings gibt es auch einige Barrieren hierbei zu überwinden (kulturelle Faktoren, unterschiedliche Vokabularien und Bezugsrahmen).[45] Als ein Instrument zur Überwindung dieser Barrieren bieten sich *Ontologien* an.[46] Das Instrument *Ontologien* stammt aus dem Feld der Forschung zur Künstlichen Intelligenz. Die meistgenannte Definition hierzu geht zurück auf Gruber (1993): eine Ontologie ist eine explizite Spe-

---

39) Siehe hierzu S. 20 f.

40) Vgl. DGQ-Band 13-11 (2001), S. 15. Die Gültigkeit als Stand der Technik ist insbesondere bei der Produkthaftung als Anerkennung durch den Gesetzgeber von Bedeutung. Ein weiterer Grund für die Verwendung der FMEA in dieser Arbeit liegt in dem Umstand begründet, dass die Kostenbeeinflussbarkeit zu Anfang des Lebenszyklusses eines Produkts signifikant am größten ist und im weiteren Verlauf abnimmt (vgl. Scheer (1992), S. 372), d. h. präventive Instrumente eignen sich in der Regel besser, um signifikant Kosten zu reduzieren (vgl. hierzu Schub, Stark (1985), S. 18, als Beispiel für den Bereich der Lebenszykluskosten von Bauobjekten).

41) Die elektronische Hilfe geht insofern über den manuellen Ansatz hinaus, als der Anwender durch die Software bereits dazu veranlasst wird, in den Kategorien der Meta-Sprache, die die Software vorgibt, zu denken. Dies erleichtert die Weiterverwendung des Wissens aus der Fehleranalyse.

42) Vgl. Keller, Kuhn (2004), S. 13; Pfeifer (2001), S. 407; Wirth, Berthold et al. (1996), S. 219.

43) Vgl. hierzu bspw. Dobry (2003), S. 1097, der auf den Unikat-Charakter von FMEAs hinweist, Gimpel, Stolten et al. (2002), S. 648, die auf die kreativen Schwierigkeiten beim Wiederauffinden von Fehlern hinweisen, und Landis, Baumann (2003), S. 992, die allgemein auf die schwere Zugänglichkeit von Wissen zur Fehlerabstellung und -vermeidung hinweisen.

44) Ähnliche Forderungen findet man auch in der Literatur: Es wird vom Nutzen gesprochen, der durch die Dokumentation von Wissen mittels FMEA erzielt werden kann (vgl. FQS (1994), S. 14).

45) Vgl. Johannsen (2000), S. 50 f. Siehe hierzu auch Jarke, Jeusfeld et al., die unterschiedliche Fachsprachen und inkompatible Systeme als Barrieren für den Wissensaustausch im Qualitätsmanagement ausmachen (vgl. Jarke, Jeusfeld et al. (1996), S. 146).

46) Im Grunde liegt – verkürzt ausgedrückt – ein Zweck dieser Arbeit in dem Nachweis des speziellen Falls, dass Ontologien für das Qualitätsmanagement geeignet sind. Für den allgemeinen Fall, dass Ontologien Barrieren überwinden helfen können, wie etwa existierende unterschiedliche Wissenshintergründe innerhalb eines Unternehmens, siehe bspw. die Ausführungen in den Kapiteln 2.1.2.4, S. 28 ff, und 4.1.4, S. 86 ff.

zifikation einer Konzeptualisierung[47] und wird in einem Wissensbasierten System genutzt, das die Möglichkeit zur Inferenz[48] besitzt. Mit Hilfe einer Ontologie lässt sich Wissen wiederverwenden und sprachliche Divergenzen lassen sich überbrücken.

Um ein Wissensbasiertes System basierend auf den Instrumenten FMEA und Ontologien zu entwickeln, bedarf es eines systematischen Vorgehens, um die Verbindung der beiden Instrumente nachvollziehbar zu realisieren. Damit ein systematisches Vorgehen sichergestellt werden kann, wird sich in der Arbeit eines weiteren Instruments bedient: ein *Vorgehensmodell*, das den Entwicklungsablauf wiedergibt und einem Benutzer ermöglicht, eine FMEA-Ontologie zu entwickeln, wird vorgestellt. Durch die Verwendung eines Vorgehensmodells wird ebenfalls eine Form der Wissenswiederverwendung sichergestellt, denn ein Vorgehensmodell zur Verbindung von Ontologien und FMEA enthält das notwendige Wissen zur Erstellung einer FMEA-Ontologie, die wiederum mittels ihrer Anwendung innerhalb eines Wissensbasierten Systems eingesetzt werden kann, um Wissen zur Fehlerreduzierung in der Produktion wiederzuverwenden.

Die Integration der drei Instrumente *Vorgehensmodell*, *Ontologien* und *FMEA* zeigt auf, wie eine hybride Wettbewerbsstrategie auf der instrumentellen Ebene realisiert werden kann.

47) Vgl. Gruber (1993), S. 200.

48) Siehe hierzu Kapitel 2.1.2.3, S. 24 ff.

1.2 Aufbau der Arbeit

Die Abbildung 3 auf Seite 14 verdeutlicht den Gang der Untersuchung in dieser Arbeit. Nach der Einführung (Kapitel 1, S. 1 ff.), die eine Abgrenzung der wissenschaftlichen Problemstellung und die besonderen Ziele der Arbeit enthält, werden die konzeptionellen Grundlagen gelegt (Kapitel 2, S. 16 ff.).

Im Anschluss wird auf die Instrumente *FMEA* (Kapitel 3, S. 35 ff.), *Ontologien* (Kapitel 4, S. 74 ff.) und *Vorgehensmodelle* (Kapitel 5, S. 120 ff.) näher eingegangen. Die Instrumente werden im Einzelnen vorgestellt und hinsichtlich ihres Umgangs mit der Ressource „Wissen" untersucht. Dabei folgt der Aufbau der drei Kapitel demselben Vorgehen, um eine systematische Nachvollziehbarkeit zu gewährleisten.

Das Kapitel 5, S. 120 ff., dient der Vorbereitung zur Entwicklung eines Vorgehensmodells für die FMEA-Ontologieerstellung. Es werden dabei die methodischen Grundlagen gelegt, indem bekannte Vorgehensmodelle aus der Literatur zu den Bereichen Software Engineering, Knowledge Engineering und Ontology Engineering hinsichtlich ihres Umgangs mit „Wissen" analysiert werden.

Kapitel 6, Seite 193 ff., umfasst ein Vorgehensmodell (das so genannte OntoFMEA-Vorgehensmodell) zur Entwicklung einer FMEA-Ontologie mittels der Verwendung von FMEA-Wissen. Das Vorgehensmodell fußt auf den Erkenntnissen der Analyse des voranstehenden Kapitels.

Mit Hilfe des vorgestellten OntoFMEA-Vorgehensmodells wird eine FMEA-Ontologie entwickelt. Das Ergebnis (die FMEA-Ontologie) wird in Kapitel 7, S. 219 ff., anhand der wichtigsten Designentscheidungen vorgestellt. Die gesamte Ontologie mit einem Ausschnitt aus einer zugehörigen Wissensbasis findet sich im Anhang A 2 der Arbeit (S. 336 ff.).

Kapitel 8, S. 252 ff., gibt ein praktisches Beispiel für den Einsatz der entwickelten FMEA-Ontologie. Es wird der Prototyp des Wissensbasierten Systems OntoFMEA vorgestellt, dessen Herzstück die FMEA-Ontologie mit ihrer Wissensbasis darstellt.

Anschließend wird in Kapitel 9, S. 279 ff., der hier vorgestellte integrierte Ansatz OntoFMEA (Vorgehensmodell, FMEA-Ontologie und Prototyp) in Hinblick auf die wissenschaftliche Problemstellung evaluiert.

Die Arbeit schließt mit einer Zusammenfassung und einem Ausblick auf zukünftige Forschungsansätze in Kapitel 10, S. 291 ff. Zusammenfassung und Ausblick ergeben sich aus den Erkenntnissen der vorangegangenen Kapitel.

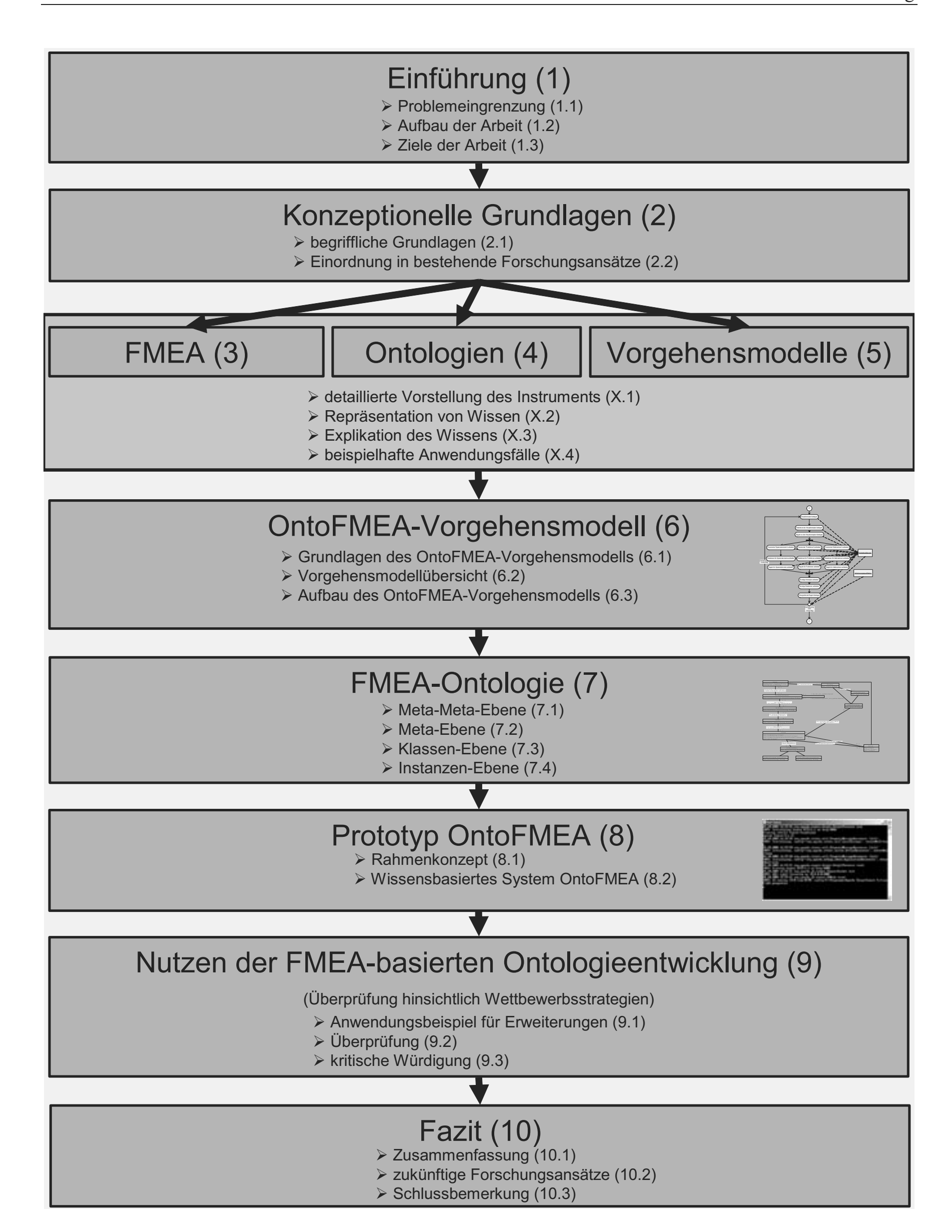


Abbildung 3: Gang der Untersuchung

1.3 Ziele der Arbeit

Es wird gezeigt, wie die Integration der Instrumente Ontologien, FMEA und Vorgehensmodelle eingesetzt werden kann, um die wissenschaftliche Problemstellung der Arbeit zu beantworten.

Die Arbeit soll einen Beitrag leisten, die Konkretisierung bei der Anwendung wirtschaftswissenschaftlicher Instrumente zur Realisierung einer hybriden Wettbewerbsstrategie (teilweise) voranzutreiben, indem sie die Integration der Instrumentenanwendung auf der Meta-Ebene des Wissensmanagements vorantreibt. Das Wissen über die Anwendung von Instrumenten auf Produktionsprozesse der Objekt-Ebene soll dabei weiter expliziert, computergestützt verarbeitbar und dauerhaft wieder verwendbar gemacht werden. Mit dem Untersuchungsgang werden fünf intendierte Ergebnisse verfolgt, die in ihrer Gesamtheit die Eignung von Erkenntnissen der Fehleranalyse zur Generierung einer FMEA-Ontologie zur Unterstützung hybrider Wettbewerbsstrategien darlegen.

Erstens wird der aktuelle *Stand der Forschung* im Bereich der Entwicklung und Anwendung von Ontologien, FMEA und Vorgehensmodellen zusammengefasst und als State-of-the-art kritisch analysiert.

Zweitens wird ein *konzeptioneller Rahmen* aufgestellt, der die Dependenzen zwischen Fehleranalyse und Ontologie(erstellung) und ontologiebasiertem Wissensmanagement aufzeigt.

Drittens wird ein *Vorgehensmodell* zur Entwicklung von Ontologien auf der Basis der Erkenntnisse der Fehleranalyse aus der FMEA entwickelt.

Viertens wird eine *FMEA-Ontologie* entwickelt, mit deren Hilfe das Wissen aus FMEA-basierten Fehleranalysen wieder verwendet werden kann.

Fünftens wird ein *Software-Prototyp* vorgestellt, der die genannte Wiederverwendung unterstützt.

Abschließend wird insbesondere die Unterstützung des entwickelten integrierten OntoFMEA-Ansatzes zur Realisierung einer simultanen Kosten- und Qualitätsführerschaft im Sinne der Konkretisierung einer hybriden Wettbewerbsstrategie gemäß der wissenschaftlichen Problemstellung dieser Arbeit dargelegt.[49]

---

49) Inwiefern die hier vorgestellten intendierten Ergebnisse zur Lösung der in Kapitel 1.1, S. 1 ff., beschriebenen wissenschaftlichen Problemstellung beitragen, wird im Kapitel 9, S. 279 ff., eingehend untersucht. Siehe hierzu insbesondere das Unterkapitel 9.2, S. 284 ff.

2 Konzeptionelle Grundlagen

Im Fokus der Arbeit stehen der Einsatz und die Dependenzen der Instrumente *FMEA* und *Ontologien*. Um die Anwendbarkeit der Erkenntnisse sicherzustellen, wird das Instrument *Vorgehensmodell* hinzugezogen. Die drei Instrumente lassen sich den Bereichen Qualitätsmanagement und Wissensmanagement zuordnen.[50] Deshalb werden die drei Instrumente hinsichtlich ihrer begrifflichen Grundlagen und ihres Forschungsstands zunächst anhand dieser beiden Bereiche erläutert.

2.1 Begriffliche Grundlagen

2.1.1 Qualitätsmanagement

2.1.1.1 Qualität

Der Begriff „Qualität" stammt aus dem 16. Jahrhundert und leitet sich vom lateinischen Wortstamm „qualis" (etwa „wie beschaffen") ab; er gilt grundsätzlich als wertneutral besetzt. In der Verwendung des Begriffs findet sich eine Vielzahl[51] unterschiedlichster Erklärungsversuche, denn „Qualität" ist kein festgelegter, einheitlicher Begriff mit klarer Bedeutung und präziser Maßeinheit, sondern vielmehr ein Wort mit vielen unterschiedlichen Begriffsinhalten.[52] Allein schon umgangssprachlich und fachsprachlich differiert der Wortgebrauch deutlich. Umgangssprachlich wird Qualität absolut begriffen, in der Regel mit der Erbringung von Bestleistungen („Wir streben nach Qualität", „Dieses Produkt[53] ist echte Qualität"). Demgegenüber hat Garvin zur Definition des Begriffs „Qualität" fünf Unterscheidungen, wie sie in der Fachsprache angetroffen werden können, herausgearbeitet:[54]

- Produktbezogener Qualitätsbegriff
  Qualität wird als Summe der vorhandenen Eigenschaften eines Produkts angesehen.

- Kundenbezogener Qualitätsbegriff
  Qualität als die wahrgenommenen Eigenschaften eines Produkts durch den Kunden.

---

50) Es sind durchaus weitere Einteilungen denkbar, jedoch erscheint es für die Bestimmung der begrifflichen Grundlagen als ausreichend, von dieser Zweiteilung auszugehen. Alternativ ließen sich bspw. Vorgehensmodelle auch dem Projektmanagement zuordnen. Weil in dieser Arbeit jedoch der Fokus auf der Wiederverwendung von Wissen beim Einsatz von Vorgehensmodelllen liegt, werden sie hier dem Wissensmanagement zugeordnet.

51) Vgl. bspw. Feigenbaum (1991), S. 3 ff.; Hennig (2001), S. 27 ff.; Huang, Lee et al. (1999), S. 43 ff.; Kocher (1989), S. 19 ff.; Pfeifer (2002), S. 5; Theden (1997), S. 6 ff.

52) Vgl. Berger (1994), S. 8.

53) In dieser Arbeit werden Sachgüter und Dienstleistungen unter dem Begriff „Produkt" subsumiert.

54) Vgl. Garvin (1984), S. 25 ff. Zollondz bezeichnet diese Unterscheidung als „einflussreich" (vgl. Zollondz (2002), S. 145). Auf die Unterscheidungen von Garvin wird bei Erklärungsversuchen zum Begriff „Qualität" oftmals Bezug genommen, so bspw. bei Hennig (2001), S. 27 f.; Theden (1997), S. 7 und Wadsworth, Stephens et al. (2002), S. 17 f.

- Qualität als absoluter Maßstab
  Qualität wird als allgemeines Maß mittels Kategorisierung in verschiedene Klassen (z. B. gut, mittel, schlecht) verwendet.

- Qualität als Vorgabe des Anbieters
  Qualität wird als Vorgabe durch den Hersteller festgelegt.

- Qualität als Wertmaßstab für Kunden
  Der Kunde schätzt das Nutzen-Preis-Verhältnis ein und bestimmt somit das Niveau der Qualität.

In den weiteren Ausführungen dieser Arbeit steht das präventive Instrument FMEA im Fokus der Betrachtungen.[55] Weiterhin wird explizit ein Produkt bei der Anwendung der FMEA in den Fokus der Betrachtungen gestellt, deshalb wird der produktbezogene Qualitätsbegriff den weiteren Ausführungen zu Grunde gelegt. Im Gegensatz zur umgangssprachlichen Verwendung findet in der Wissenschaft beim Versuch, den produktbezogenen Begriff „Qualität" verbindlich zu definieren, die DIN EN ISO 8402 weite Verbreitung[56]:

Qualität ist die Gesamtheit von Merkmalen eines Produkts bezüglich ihrer Eignung, festgelegte und vorausgesetzte Anforderungen zu erfüllen.[57]

Qualität lässt sich im Sinne dieser Definition als Ausmaß der Anpassung des Ergebnisses einer Tätigkeit an die vorgegebenen Anforderungen sehen.[58] Hierbei bedeutet Qualität nicht

55) Siehe zu weitergehenden Erläuterungen Kapitel 3.1, S. 35 ff.

56) Vgl. Hennig (2001), S. 27; Ruta (1999), S. 34; Theden (1997), S. 8; Zollondz (2002), S. 152.

57) DIN EN ISO 8402:1995, Abs. 2.1. Der Begriff „Erfordernisse", wie er sich ursprünglich in der zitierten Definition findet, wird im Folgenden mit dem Begriff „Anforderungen" ersetzt. Der ursprünglich in der Definition zu findende Begriff „Einheit" wird im Folgenden durch den Begriff „Produkt" ersetzt, damit der Textkontext konsistent bleibt.
Eine ähnlich lautende Definition für Qualität findet sich in DIN EN ISO 9000:2000, Abs. 3.1.1: Qualität ist der Grad, in dem ein Satz inhärenter Merkmale Forderungen erfüllt. Auch der Begriff „Forderungen" wird im Folgenden durch den Begriff „Anforderungen" ersetzt. Nach der Fertigstellung eines Produkts gehören die inhärenten Merkmale als ständige qualitätsbezogene Merkmale zu einem Produkt (bspw. *Zuverlässigkeit* und *Normgerechtigkeit*; vgl. zu weitergehenden Ausführungen zum Begriff „inhärent" DGQ-Band 11-04 (2002), S. 52, Zollondz (2002), S. 154, und zu einer Auflistung von qualitätsbezogenen Merkmalen Garvin (1987), S. 104 ff., und Garvin (1988), S. 68 ff.).

58) Hierüber lässt sich trefflich streiten, denn Anforderungen müssen nicht nur, wie in dieser ingenieurtechnisch geprägten Sichtweise von außen vorgegeben sein, sondern können auch von innen „aktiv" gestaltet werden (Management geprägte Sichtweise). Weil jedoch die Anwendung der FMEA in ihrer Grundkonzeptionierung eine von außen geprägte Sichtweise einnimmt, wird für die Definitionsfindung ebenfalls eine Sichtweise von außen eingenommen.

bloß die Konformität des Produkts mit seinen technischen Spezifikationen, sondern beinhaltet die weitergehende Erfüllung der Kunden- und Nutzeranforderungen.[59]

2.1.1.2 Qualitätsmanagementsysteme

Um im Unternehmen die im vorherigen Kapitel definierte Qualität eines Produkts sicher zustellen, bedarf es innerhalb eines Unternehmens des systematischen Einsatzes der betrieblichen Mittel, d. h., innerhalb des Unternehmens müssen sämtliche qualitätsbezogenen Tätigkeiten „gemanaged“ werden. In Anlehnung an die DIN EN ISO 9000:2000 wird im Sinne einer Arbeitsdefinition Qualitätsmanagement als die aufeinander abgestimmten Tätigkeiten zum Leiten und Lenken eines Unternehmens bezüglich der Qualität verstanden.[60]

Dabei orientiert sich das Qualitätsmanagement allgemein an den klassischen Managementfunktionen[61] (einschließlich der Erweiterung dieser Funktionen) und ist als Gesamtheit aller qualitätsbezogenen Tätigkeiten und Zielsetzungen in folgende Funktionen gegliedert:[62]

- Qualitätsplanung
  Auswahl, Klassifizierung und Gewichtung der Qualitätsmerkmale sowie die Konkretisierung der Qualitätsanforderungen unter Berücksichtigung von Anspruchsniveau und Realisierungsmöglichkeiten vor Beginn der Produktion gehören zur Qualitätsplanung.

- Qualitätslenkung
  Die Qualitätslenkung umfasst die Überwachung und Korrektur der Realisierung eines Produkts mit dem Ziel, die Qualitätsanforderungen zu erfüllen. Dabei werden die erzielten Ergebnisse mit den Vorgaben aus der Qualitätsplanung abgeglichen.

- Qualitätsprüfung
  Mit der Qualitätsprüfung wird die Kontrolle, inwieweit die definierten Qualitätsanforderungen erfüllt wurden, durchgeführt. Ermittelt werden die Ist-Werte der Produkt- und Prozessqualität.

- Qualitätsverbesserung
  Die Entwicklung des Qualitätsgedankens soll im Unternehmen stetig vorangebracht

59) Vgl. Pfeifer (2002), S. 5. In diesem Sinne werden die beiden Unterscheidungen von Garvin „produktbezogen“ und „kundenbezogen“ letztlich zu einer Definition von Qualität vereint. Der produktbezogene Qualitätsbegriff lässt sich auch als technische Qualität und der kundenbezogene Qualitätsbegriff als wahrgenommene Qualität verstehen, die letztlich dasselbe Produkt referenzieren. Juran prägte in diesem Zusammenhang die Formulierung „fitness for use“, die ein Produkt als zentrale Anforderung gestellt bekommt (vgl. Zollondz (2002), S. 85). Zu den Modellen, die das Konstrukt Kundenzufriedenheit zur Messung der Kundenanforderungen beschreiben, gehört bspw. das C/D-Modell (Confirmation/Disconfirmation-Paradigm; vgl. Bahr (1999), S. 43).

60) Vgl. DIN EN ISO 9000:2000, Abschnitt 3.2.8.

61) Gemäß Koontz und O´Donnell lassen sich als klassische Managementfunktionen Planung, Organisation, Personaleinsatz, Kontrolle und Leitung identifizieren (vgl. Koontz, O´Donnell (1972), S. 1 ff.).

62) DGQ-Band 11-04 (2002), Abschnitt 2.2.6. Gemäß DIN EN ISO 9000:2000, Abschnitt 3.2.8, umfasst das Leiten und Lenken bezüglich der Qualität üblicherweise zu den folgend genannten Teilen zusätzlich noch die Festlegung der Qualitätspolitik und der Qualitätsziele.

werden. Ziel der Funktion Qualitätsverbesserung ist es, eine Struktur der ständigen Verbesserung zu erhalten.

Der zu Anfang dieses Kapitels geforderte *systematische* Einsatz von qualitätsbezogenen Tätigkeiten wird unter dem Begriff Qualitätsmanagementsystem zusammengefasst. Ein Qualitätsmanagementsystem entspricht einem Satz von in Wechselbeziehung stehenden Elementen zum Leiten und Lenken eines Unternehmens bezüglich der Qualität; die Elemente legen Politik und Ziele fest und dienen dem Erreichen dieser Ziele.[63]

Zum Lenken von Qualität im Unternehmen wurden qualitätsspezifische Instrumente entwickelt, die zusammen angewendet ein Qualitätsmanagementsystem unterstützen. Zu diesen Instrumenten im engeren Sinne (aufgrund ihres direkten Bezugs zum Qualitätsmanagement) werden i. d. R. gezählt:[64]

- Q7 (7 Qualitätswerkzeuge)[65],
- M7 (7 Managementwerkzeuge)[66],
- QFD (Quality Function Deployment)[67],
- FMEA (Fehlermöglichkeits- und Einflussanalyse)[68],
- DoE (Design of Experiments)[69] und
- SPC (Statistical Process Control)[70].

Zu den Instrumenten im weiteren Sinne (d. h. ihre ursprüngliche Verwendung lag nicht im Bereich der „Qualität“) lassen sich bspw. Brainstorming, Delphi-Methode, Morphologischer Kasten und Synektik zählen.[71]

Es ist bei den Instrumenten möglich, diese isoliert zu verwenden, um Teilziele zu erreichen. Einige können auch ergänzend zueinander verwendet werden, wie bspw. FMEA und QFD.

---

63) Vgl. DIN EN ISO 9000:2000, Abschnitte 3.2.1, 3.2.2 und 3.2.3.

64) Vgl. Zollondz (2002), S. 319 f.

65) Vgl. Ruta (1999), S. 35 ff.

66) Vgl. Gogoll (1994), S. 516 ff.

67) Vgl. Pfeifer (2002), S. 303 ff.

68) Siehe hierzu Kapitel 3, S. 35 ff.

69) Vgl. Pfeifer (2002), S. 343 ff.

70) Vgl. Wadsworth, Stephens et al. (2002), S. 125 ff.

71) Vgl. Zollondz (2002), S. 319 f. Siehe ebenda für nähere Erläuterungen zu diesen Instrumenten. Alternativ siehe auch Pfeifer (2002), S. 275 ff.

Die meisten Instrumente sind einer Phase im Lebenszyklus eines Produkts zugeordnet oder sie lassen sich Grundelementen eines Qualitätsmanagementsystems zuordnen.[72]

2.1.1.3 Fehler

Es wird in dieser Arbeit auf das Instrument FMEA näher und deshalb auch auf die hiermit verknüpfte Verwendungsweise des Begriffs „Fehler" eingegangen. Die FMEA nimmt hinsichtlich ihrer Anwendungshäufigkeit eine herausragende Stellung unter den genannten Instrumenten ein. So ermittelt Theden in einer Umfrage unter 93 Unternehmen, dass in 77,4% der Unternehmen die FMEA eingesetzt wird.[73] Allerdings erscheint die Stichprobe nur eingeschränkt repräsentativ, da sie nicht auf die Gesamtheit aller Unternehmen in Deutschland bezogen werden kann, weil zu Anfang überprüft wurde, ob die Unternehmen in der Lage sind, eine Aussage zu Instrumenten des Qualitätsmanagements zu machen.[74] Wird jedoch berücksichtigt, dass für zertifizierte Unternehmen des Automobilbaus die Anfertigung einer FMEA zwingend vorgeschrieben wird und im Jahr 2003 ca. 70% der kleinen und mittelgroßen Unternehmen in Deutschland (≤ 500 Mitarbeiter) und 81% der Großunternehmen (>500 Mitarbeiter) in Deutschland eine DIN-ISO-Zertifizierung innehielten[75], dann können die genannten Daten als Indikator für die Wichtigkeit einer FMEA herangezogen werden, denn für die anderen Instrumente wird eine Verwendung nicht explizit gefordert.[76] Ähnlich argumentierend, kennzeichnet Pfeifer die FMEA als einzige etablierte Technik des Qualitätsmanagements, die in 75% der produzierenden Unternehmen Verwendung findet.[77]

Zentraler Bestandteil bei der Durchführung einer FMEA ist eine Fehleranalyse. Um das Ziel „Qualität" zu erreichen, müssen Abweichungen von festgelegten oder vorausgesetzten Anforderungen an ein Produkt vermieden werden. Für den Kunden äußern sich dabei Abweichungen von Anforderungen als Fehler. Hieraus ergibt sich gemäß DIN EN ISO 9000:2000, dass ein Fehler als die Nichterfüllung einer Anforderung definiert wird.[78]

Im technischen Bereich kann von einem Fehler gesprochen werden, wenn Merkmalswerte von durch Sollwerte festgelegten Anforderungen bei einem Produkt abweichen. Dabei wird

72) Mögliche Grundelemente eines Qualitätsmanagementsystems sind bspw. *Prüfmittelüberwachung*, *Prozesslenkung* und *Fehlerdiagnose*. Zur näheren Erläuterung der Grundelemente eines Qualitätsmanagementsystems siehe VDI/VDE 2621, S. 10 ff.

73) Vgl. Theden (1997), S. 87. Nur den Bereich der Automobilindustrie isoliert betrachtet, findet sich sogar ein Wert von 96% Einsatzhäufigkeit (vgl. Theden (1997), S. 88).

74) Vgl. Theden (1997), S. 83.

75) Vgl. ExBa (2003), S. 36.

76) Ähnlich argumentiert auch der VDA, der bereits 1986 bei 96% der in der Automobilindustrie tätigen Unternehmen die FMEA im Einsatz sieht (vgl. VDA (1986), S. 29 ff.). Leider bleibt bei der Quelle jedoch unklar, woher diese Zahl stammt, d. h. eine Nachprüfbarkeit ist nicht gegeben.

77) Vgl. Pfeifer (2002), S. XXVII.

78) Vgl. DIN EN ISO 9000:2000, Abschnitt 3.6.2. Eine Anforderung wird in dieser Arbeit inhaltlich identisch gesetzt mit einem Erfordernis oder einer Erwartung, das oder die festgelegt, üblicherweise vorausgesetzt oder verpflichtend ist. Dies entspricht einer an die DIN angepassten Verwendungsweise (vgl. DIN EN ISO 9000:2000, Abschnitt 3.1.2).

unter einem Merkmal eine Eigenschaft verstanden, die einem Produkt inhärent oder zugeordnet und qualitativ oder quantitativ sein kann.[79]

Weil für die FMEA die Fehlerfolge, d. h. der Einfluss einer vorliegenden Nichterfüllung einer Anforderung, einen besonderen Stellenwert einnimmt, wird gemäß der Richtlinie VDI 3822-1:1984[80] unterschieden in Fehler, die einen

- Vorschaden,
- Primärschaden,
- Folgeschaden oder
- Wiederholungsschaden

als Fehlerfolge bedeuten.

Ein Vorschaden stellt einen früher, d. h. vor dem Zeitpunkt der Betrachtung, aufgetretenen Schaden dar. Ein Primärschaden gilt als zeitlich zuerst, d. h. nach dem Zeitpunkt der Betrachtung, aufgetretener Schaden. Ein Primärschaden kann wiederum Ursache (als Vorschaden) für weitere Schäden (Folgeschäden) sein.[81]

79) Vgl. DIN EN ISO 9000:2000, Abschnitt 3.5.1.

80) In VDI 3822-1:1984 wird der Begriff „Schadensanalyse“ ins Englische übersetzt mit „failure analysis“. Linguistisch korrekt erfolgt die Übersetzung von „failure“ in DIN 25448:1990 mit „Ausfall“. Allgemein wird jedoch in der aktuellen deutschsprachigen Literatur das Englische „failure“ mit „Fehler“ übersetzt (vgl. bspw. DGQ-Band 13-11 (2001), S. 5 und S. 9; Stockinger (1989), S. 155). Eberhardt spricht in diesem Zusammenhang sogar von einer geglückten Übersetzung (vgl. Eberhardt (2003), S. 17). Im Folgenden wird sich der Mehrzahl der Verwendungen angeschlossen, um das Verständnis für den Leser zu erleichtern, d. h. die Begriffe „Fehler“, „Ausfall“ und „Schaden“ werden synonym verwendet.

81) Vgl. VDI 3822-1:1984, S. 2.

2.1.2 Wissensmanagement

2.1.2.1 Wissen

Wissensmanagement hat die Bewirtschaftung des Produktionsfaktors Wissen zum Ziel.[82]

Zur Verwendung dieser Definition muss zunächst der Begriff „Wissen" weiter erläutert werden.[83] Bei der Spezifizierung des Begriffs hilft die problemorientierte Definition von Probst, Raub und Romhardt:[84]

Wissen entspricht der Gesamtheit der Kenntnisse und Fähigkeiten, die Akteure zur Lösung von Problemen einsetzen können.

Die in dieser Definition für Wissen benutzten Begriffe, „Kenntnisse" und „Fähigkeiten", basieren auf einer Einteilung in zwei Wissensrepräsentationsextreme, die ihren Ursprung in einer Unterscheidung von Ryle[85] (Knowing-that/Wissen-Was und Knowing-how/Wissen-Wie) haben.

Das Wissen-Was des Akteurs entspricht deklarativem (Fakten- und Fach-)Wissen. Es umfasst die Kenntnisse eines Akteurs über Objekte und ihre Eigenschaften sowie Relationen der Objekte untereinander. Die deklarative Wissensrepräsentation lässt sich als Sammlung statischer Elemente verstehen. Das beschriebene Wissen kann hierarchisch oder netzwerkartig strukturiert sein. Vorteilhaft sind einfaches Ergänzen, Ändern und Entfernen von Elementen der Repräsentation. Nachteilig erscheinen jedoch die begrenzten Darstellungsmöglichkeiten, da bspw. eine chronologische Verknüpfung nur schwer möglich wird.

Demgegenüber steht das Wissen-Wie des Akteurs, das einen prozeduralen Charakter aufweist, denn es umfasst das Wissen über Tätigkeiten (Aktivitäten). Die prozedurale Wissensrepräsentation lässt sich als Sammlung dynamischer Elemente zur Beschreibung von Hand-

82) Gerst, Hackl et al. (2001), S. 51.

83) Bei der Definition des Begriffs „Wissen" steht man vor dem Problem, dass es sich bei „Wissen" um einen selbstbezüglichen Begriff handelt: der Versuch der Definition des Begriffs „Wissens" verändert diesen Begriff im Moment des Versuchs, weil die Definition selbst Bestandteil des Wissens wird. So lässt sich in der Literatur keine allgemein zufrieden stellende Definition für den Begriff finden. Aufgrund dieser anfänglichen Unmöglichkeit wird in dieser Arbeit auf eine Untersuchung von Definitionsansätzen verzichtet und gleich auf eine für diese Arbeit gültige Definition eingegangen. Der Verfasser ist sich dabei bewusst, dass sich auch diese Definition je nach Betrachtungsstandpunkt angreifen lässt, er erachtet sie jedoch für die vorliegende Problemstellung als ausreichend. Siehe zur Problematik der Begriffsdefinition von „Wissen" bspw. Obrst, Liu (2003), S. 104 f.; Pocsai (2000), S. 18 ff., und Puppe, Stoyan et al. (2003), S. 599 f.

84) Vgl. Probst, Raub et al. (2003), S. 23. Auf diesen Definitionsversuch wird insbesondere zurückgegriffen, weil die zentrale Aufgabe eines Wissensbasierten Systems (siehe hierzu Kapitel 2.1.2.3, S. 24 ff.) in der Repräsentation und Verarbeitung von problembezogenem Wissen liegt (vgl. Beierle, Kern-Isberner (2003), S. 7).

85) Vgl. Ryle (1949), S. 25 ff. Auch wenn Baier in Ryle (1969), S. 26, als Übersetzer auf die Schwierigkeit der Sinn erhaltenden Übersetzung von Knowing-how und Knowing-that hinweist, wird im Text eine eigene Übersetzung vorgenommen im Sinne einer Arbeitsdefinition. Sie dient an dieser Stelle lediglich der Verdeutlichung.

lungen verstehen. Es umfasst das Wissen eines Akteurs[86] über „Anwendungshinweise" des Wissens bereits in der Darstellung selbst.[87] Es werden mit Hilfe des prozeduralen Wissens, aufbauend auf deklarativem Wissen, Aufgaben gelöst.[88] Der Gebrauch des Wissens steht somit im Vordergrund dieser Repräsentationsform.

Lässt sich deklaratives oder prozedurales Wissen sprachlich repräsentieren, so lässt es sich auch als Modell darstellen, das in einem nächsten Schritt formalisiert und in einem Computer repräsentiert werden kann.[89] In diesem Sinne lässt sich Wissen für den Kontext dieser Arbeit unter Einbezug der vorigen Aussagen in einer Arbeitsdefinition erweitern zu der Menge aller Aussagen, die über einen repräsentierten (Welt-)Ausschnitt gemacht und zur Problemlösung eingesetzt werden können.[90]

2.1.2.2 Wissensrepräsentation

Nachdem Wissen akquiriert worden ist, stellt sich die Problematik, wie es repräsentiert werden soll, damit es genutzt und erhalten werden kann. Zusätzlich stellt sich die Problematik, wie weiteres Wissen zu repräsentiertem Wissen hinzugewonnen werden kann.

Als Wissensrepräsentation im Bereich der KI wird die sprachliche Rekonstruktion von Wissen und ihre Implementierung verstanden.[91] Die sprachliche Rekonstruktion umfasst die Beziehung eines Begriffs zu dessen semantischer/pragmatischer Bedeutung als Objekt in der Realität oder Fiktion. Die Implementierung umfasst die rechnergestützte Nutzbarmachung des zu repräsentierenden Wissens. Die rechnergestützte Nutzbarmachung von Wissen erfolgt mittels eines Wissensbasierten Systems.[92] Die Wissensrepräsentation findet dabei ihre (praktische) Verwendung innerhalb eines Wissensbasierten Systems.

Wissen soll so dargestellt werden, dass eine „intelligente" Maschine zu neuen „Erkenntnissen" durch formale Manipulation des repräsentierten Wissens kommen kann.[93] Die Darstel-

86) Akteure können sowohl natürliche Personen als auch Unternehmen oder Maschinen(systeme) sein.

87) Strube, Habel et al. weisen in diesem Zusammenhang darauf hin, dass dieses Handlungswissen nicht immer in allen Fällen sprachlich repräsentierbar ist (Strube, Habel et al. (2003), S. 38).

88) Vgl. Pocsai (2000) S.18.

89) Dieser Aussage liegt eine Spielart der starken KI-These als Argumentationsprämisse zu Grunde: Alles, was sich sprachlich artikulieren lässt, kann auch formalsprachlich repräsentiert werden. Eine ähnliche Sichtweise findet sich bspw. in Moravec (1990), S. 9 ff. Kritik an dieser Prämisse wird, wenn auch nicht immer ganz sachlich, bspw. von Eurich (1988), S. 79 ff., und Weizenbaum (1977), S. 268 ff., geübt. Ein informativer Diskurs in Form eines Streitgesprächs zwischen Joseph Weizenbaum und Klaus Haefner findet sich in Haller (1990), S. 59 ff.

90) Vgl. Haun (2000), S. 33.

91) Vgl. Owsnicki-Klewe, von Luck et al. (2003), S. 153. An dieser Stelle erfolgt eine Fokussierung der Verwendung des Begriffs „Wissensrepräsentation" auf den Bereich der KI-Forschung, weil später in der Hauptsache auf Ontologien aus der KI-Forschung eingegangen wird. Im Sinne dieses eingegrenzten Verständnisses von „Wissensrepräsentation" sind natürlichsprachliche Texte nur dann Wissensrepräsentationen, wenn dieses Wissen mittels eines Formalismus computergestützt verwendet werden kann.

92) Siehe nächstes Kapitel, S. 24.

93) Vgl. Brachman, Levesque (1985), S. XIII.

lungsart *formal* ist kennzeichnend für den Bereich der Wissensrepräsentation innerhalb der KI-Forschung.[94]

Der Wissensrepräsentation wird im Folgenden die Aufgabe der Rekonstruktion und rechnergestützten Nutzbarmachung von explizitem und implizitem Wissen zugeschrieben.[95]

Der Vorgang, der die Anwendung von Logik[96] und Ontologie für die Aufgabe der Erschaffung lauffähiger Computermodelle als Wissensrepräsentation vorsieht, wird als Knowledge Engineering bezeichnet.[97]

2.1.2.3 Wissensbasierte Systeme

Ein Wissensbasiertes System (WBS)[98] wird als informationstechnisches System[99] definiert, das das Problemlöseverhalten (eines Experten) nachbilden soll.[100] Die Verarbeitung von Wis-

---

94) Vgl. Bibel (1993), S. 13 f.

95) In der Literatur wird explizites Wissen als Wissen definiert, das Akteure problemlos kommunizieren und schriftlich dokumentieren können. Hingegen wird implizites Wissen, das häufig mit tazitem Wissen („tacit knowledge") gleichgesetzt wird, als Wissen definiert, über das Akteure verfügen. Es lässt sich jedoch nicht problemlos kommunizieren. Als Beispiel für implizites Wissen wird in der Literatur häufig angeführt, dass ein Akteur Fahrrad fahren kann, ohne erläutern zu können, welche Handlungen er ausführt, wenn er Fahrrad fährt. Vgl. zu diesem Begriffsverständnis Nonaka, Takeuchi (1995), S. 8 und ferner z. B. Grant (2002), S. 177 f.; Mohr (1999), S. 10 f.; Schindler (2001), S. 30 ff.; Willke (2001), S. 12 ff. Diesem Begriffsverständnis wird in der Arbeit nicht gefolgt, da es zum einen aufgrund der originären Semantik des Begriffs „implizit" irreführend ist und zum anderen im Hinblick auf die Computerverarbeitung als nicht Ziel führend angesehen werden kann. Explizites Wissen wird deshalb als Wissen verstanden, welches in Form von Dokumenten mittels eines Formalismus verarbeitbar vorliegt. Implizites Wissen wird als Wissen verstanden, das explizitem Wissen immanent ist, jedoch nicht ohne weiteres Handeln dem Computer, der einen Formalismus anwendet, als solches zur Verfügung steht. Das heißt, implizites Wissen muss erst noch expliziert werden. Als Beispiel sei angeführt: In einem Dokument ist vermerkt, dass eine Autorin X für dieses Dokument genannt wird. Als explizites Wissen liegt vor: „Die Autorin des Dokuments ist X." Implizites Wissen wäre zum Beispiel, dass die Autorin eine Frau ist. Nach diesem Begriffsverständnis steht explizites und implizites Wissen den Akteuren eines gemeinsam verwendeten Weltausschnitts zur Verfügung und wird unter dem Begriff organisationales Wissen subsumiert. Dem organisationalen Wissen wird hier das subjektgebundene Wissen gegenübergestellt, welches das tazite und das explizierbare Wissen umfasst. Dieses Begriffspaar bildet zusammen das Wissen eines Akteurs, wobei tazites Wissen dem Begriff des impliziten Wissens in der Literatur entspricht (Fahrradfahrbeispiel). Das explizierbare Wissen ist das Wissen, welches der Akteur im Kopf hat und problemlos kommunizieren und schriftlich dokumentieren könnte.

96) Der Begriff Logik wird in dieser Arbeit oftmals abkürzend benutzt, wenn tatsächlich ein logisches Kalkül gemeint wird (siehe hierzu auch Kapitel 4.2.1.2, S. 98 ff.).

97) Vgl. Sowa (2000), S. 132. Der Begriff *Knowledge Engineering* hat sich auch im deutschsprachigen Raum durchgesetzt, deshalb wird auf eine Übersetzung an dieser Stelle verzichtet. Der Begriff *Ontologie* wird an dieser Stelle noch im philosophischen Sinne verwendet.

98) Der Begriff *Expertensystem* wurde in neuerer Zeit durch den Begriff *Wissensbasiertes System* ersetzt (vgl. bspw. Beierle, Kern-Isberner (2003), S. 10; Schreiber, Akkermans et al. (2001), S. 23). Ursprünglich sollte das Wissen von Experten zu einem Sachgebiet computerverarbeitbar dargestellt werden. In jüngerer Zeit gelangten jedoch auch Ansätze, die bspw. allgemeines Weltwissen darstellen, zum Einsatz. Die beiden Begriffe werden in dieser Arbeit synonym verwendet, weil auch umfassendes allgemeines Weltwissen als eine spezifische Art von Expertenwissen aufgefasst werden kann.

99) Ein informationstechnisches System entspricht einem Computersystem bestehend aus Hard- und Software.

100) Vgl. VDI/VDE 2621, S. 50.

sen mittels Computer steht im Fokus des Einsatzes von Wissensbasierten Systemen. Wissensbasierte Systeme bestehen in der Regel aus den Komponenten *Wissensakquisition*, *Dialog*, *Erklärung*, *Inferenzkomponente/(-maschine*[101]*)* und *Wissensbasis* (siehe Abbildung 4). Den Kern eines WBS bilden die Wissensbasis und die Inferenzkomponente.

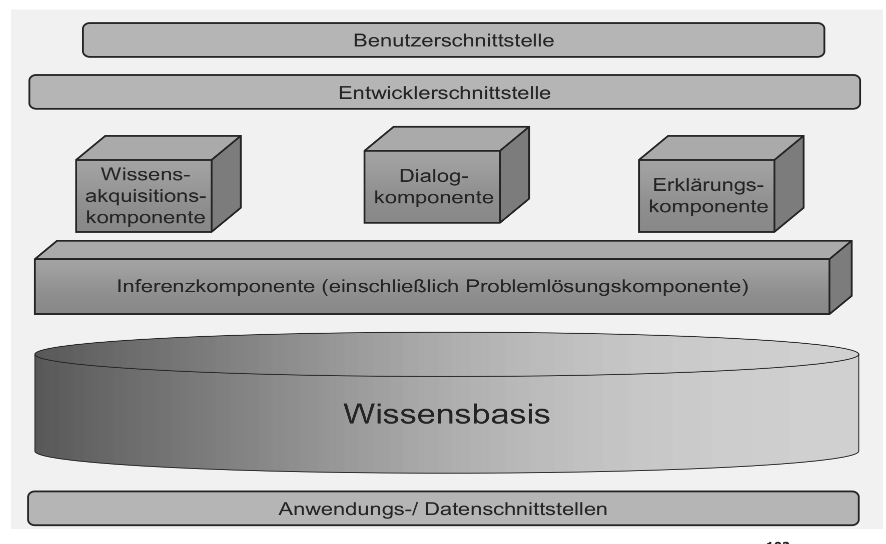


Abbildung 4: Schematischer Aufbau eines Wissensbasierten Systems[102]

Die Aufgaben der einzelnen Komponenten ergeben sich wie folgt:

- Wissensbasis
  Als Komponente eines WBS dient die Wissensbasis zur formalen Rekonstruktion (Repräsentation) von Wissen. Üblicherweise enthält die Wissensbasis während der Anwendung drei Arten von Wissen:[103]

  - *implementierungsspezifisches Wissen*, welches vor der Anwendung akquiriert und repräsentiert wurde und sich bei der Anwendung nicht ändert,

---

101) Die Begriffe Inferenzkomponente und Inferenzmaschine werden in der Arbeit synonym verwendet.

102) Um den schematischen Aufbau eines WBS zu verdeutlichen, wurde in der Abbildung auf die Darstellung der funktionalen Zusammenhänge zwischen den Komponenten verzichtet. Ähnliche Aufbauten, die sich lediglich in Nuancen unterscheiden und teilweise auch die funktionalen Zusammenhänge beinhalten, lassen sich finden in Durkin (1998), S. 4.2 (hier wird auf die Komponente *Wissensakquisition* verzichtet und zusätzlich die Komponente *Arbeitsspeicher* für das Ablegen von Zwischenergebnissen eingeführt); Haun (2000), S. 126 ff. (hier wird anfänglich zwischen Inferenzkomponente und Problemlösungskomponente unterschieden, auf S. 134 räumt der Autor jedoch ein, dass „Inferenzverfahren und Problemlösungsmethoden heute eine Einheit bilden"); Struckmann (2002), S. 55 (hier wird zusätzlich in Anwendungs- und Datenschnittstellen unterschieden, die in der hier vorliegenden Abbildung zusammengefasst dargestellt werden), sowie Beierle, Kern-Isberner (2003), S. 17; Spur, Krause (1997), S. 319.

103) Vgl. Struckmann (2002), S. 53 f.

- *anwendungsspezifisches Wissen*, welches vom Benutzer im Laufe der Anwendung eingegeben wurde, und
- *expliziertes Wissen*, welches aus Zwischen- und Endergebnissen besteht, das vom System während der Anwendung abgeleitet, d. h. aus implizitem Wissen expliziert, wurde.[104]

- Inferenzkomponente
  Mittels Schlussfolgerungen wird in der Inferenzkomponente implizit in der Wissensbasis enthaltenes Wissen expliziert.[105] Mittels der Inferenzkomponente werden aus der Wissensbasis „gesuchte" Ergebnisse häufig mit Hilfe von Regeln abgeleitet.[106]

- Erklärungskomponente
  Hier wird dem Anwender ermöglicht, die Ermittlung von Wissen durch die Inferenzkomponente nachzuvollziehen.[107]

- Schnittstellen
  Es lassen sich drei Arten von Schnittstellen differenzieren.[108] Die *Benutzerschnittstelle*, *Entwicklerschnittstelle* und *Anwendungs-/Datenschnittstellen.* Die Benutzerschnittstelle ermöglicht die Kommunikation des Benutzers mit dem WBS. Die Entwicklerschnittstelle ermöglicht dem Wissensingenieur[109] neben anderen Aufgaben, wie bspw. der Entwicklung neuer Funktionalitäten, das Einpflegen zusätzlicher Wissensbestände. Mit Hilfe der Entwicklerschnittstelle wird die Weiterentwicklung des Systems vollzogen.[110]

104) Zu den Möglichkeiten der Wissensrepräsentation siehe Kapitel 4.2, S. 95 ff.

105) Diese Schlussfolgerungen basieren auf semantischen *Regeln*, die zur Anwendung kommen. Eine mögliche Form dieser Regeln stellen semantische *Inferenzregeln* dar. Mit Hilfe dieser Regeln wird festgelegt, wie implizites Wissen durch Schlussfolgern expliziert werden kann. Es lässt sich dabei zwischen deduktiven und non-deduktiven semantischen Inferenzregeln unterscheiden. Die *deduktiven Regeln* nehmen als formal-logische Inferenzregeln auf die äußere Gestalt – die z. B. in prädikatenlogischer Form vorliegt – des expliziten Wissens Bezug. Die Deduktion wird dabei mit der Annahme ausgeführt, dass es sich immer um korrekte Schlussfolgerungen handelt. Das abgeleitete Wissen ist stets wahr, bspw. wenn vom allgemeineren Fall auf einen speziellen Fall geschlossen wird (vgl. Beierle, Kern-Isberner (2003), S. 23). Im Gegensatz hierzu stehen die *non-deduktiven Regeln*. Die non-deduktiven Regeln des inhaltlichen oder natürlichen Schließens werten vornehmlich die Bedeutungen der natürlichsprachlichen Ausdrücke aus (Abduktion und Induktion). Charakteristisch hierfür ist, dass keineswegs immer korrektes Wissen hergeleitet wird, bspw. wenn von einem speziellen Fall auf das Allgemeine geschlossen wird. Weitere semantische Regeln lassen sich als *Integritätsregeln* zur Einschränkung der Interpretations- und Verknüpfungsmöglichkeiten der Begriffe bestimmen. Sie legen die Verknüpfungen natürlichsprachlicher Ausdrücke fest, die – über die syntaktisch korrekte Verknüpfung hinaus – inhaltlich zulässig sind.

106) Vgl. Puppe (1991), S. 2; Spur, Krause (1997), S. 318; Vámos (1998), S. 3.7.

107) Bei einem regelbasierten System geschieht dies bspw. durch Aufzählung der angewendeten Regeln und der zugehörigen Zwischenergebnisse.

108) Vgl. Durkin (1998), S. 4.2; Struckmann (2002) S. 55.

109) Zum Begriff des „Wissensingenieurs" wird VDI/VDE 2621, S. 50, gefolgt: Der Wissensingenieur ist ein Fachmann, dessen Aufgabe ist, das Wissen eines Experten in einer dem verwendeten WBS entsprechenden Form der Wissensrepräsentation darzustellen.

110) Neben der direkten Bearbeitung der Wissensbasis per „harter" Eingabe von Hand gehört hierzu bspw. auch die Umsetzung eines verbesserten Suchalgorithmus für die Inferenzkomponente.

Die Anwendungs-/Datenschnittstellen vernetzen das System mit weiteren Quellen, die das WBS in der Funktionserfüllung unterstützen.[111]

- Dialogkomponente
  Die Dialogkomponente vollzieht die Kommunikation mit dem Anwender des Systems. Dabei greift sie auf entsprechendes Dialogwissen der Wissensbasis zurück und stellt dieses dem Anwender in Form der Unterstützung der Gestaltung einer „intelligenten" Kommunikation zur Verfügung.[112]

- Wissensakquisitionskomponente
  Die Wissensakquisitionskomponente[113] dient der Manipulation der Wissensbasis im gewählten Repräsentationsformat. Die Wissensakquisition hat die Aufgabe, Wissen zu finden, einzugrenzen, zu erfassen und zu formalisieren.[114] Zwei Extreme der Wissensakquisition sind hierzu denkbar. Zum einen kann die Akquisition über die Entwicklerschnittstelle durch den Wissensingenieur (bspw. mittels eines Editors) quasi manuell geschehen. Zum anderen kann sie vollautomatisch ohne menschliche Hilfe durch unterstützende Anwendungssoftware vollzogen werden.[115]

Wissensbasierte Systeme der so genannten 1. Generation beschrieben den Problemraum ausschließlich mit Hilfe von Fakten und Regeln. Ein Schlussfolgerungsmechanismus wertete diese Regeln aus.[116] Der ursprüngliche Transferansatz bei der Wissensakquisition, der eine direkte unmittelbare Rekonstruktion des Wissens des Experten durch den Programmierer vorsah, stellte sich jedoch aufgrund mangelnder Eignung zur langfristigen Wartung großer Wissensbasen als nicht ausreichend heraus.[117] Hieraus entwickelte sich der so genannte Modellbasier-

---

111) Die Schnittstellen ermöglichen bspw. den Zugriff auf bestehende Dateien oder Datenbanken (Datenschnittstellen) oder sie übergeben Informationen an weitere Anwendungen aus dem WBS heraus (Anwendungsschnittstellen) (vgl. Struckmann (2000), S. 54).

112) Bspw. kann sie dem Anwender eine Baumstruktur zur Verfügung stellen, damit dieser bei einer semantischen Suche Erleichterung erfährt.

113) Abweichend findet sich in der Literatur auch die gleichbedeutende Bezeichnung „Wissenserwerbskomponente" (vgl. Beierle, Kern-Isberner (2003), S. 17; Puppe (1991), S. 13).

114) Vgl. Haun (2000), S. 133.

115) Während die erste Möglichkeit unter den üblichen „Schwachstellen" menschlicher Existenz leidet (Subjektivität, begrenzte Kapazität, geringe Geschwindigkeit usw.), gibt es für die letztere Möglichkeit (noch) keine Anwendungen, die nicht ebenfalls unter „Schwachstellen" gleichwohl künstlicher Existenz leiden (lediglich symbolhaftes Verständnis, mangelnde Abstraktionsfähigkeit, beschränkte Einsatzmöglichkeiten aufgrund proprietärer Systeme usw.).

116) Vgl. Krallmann (1992), S. 218.

117) Vgl. Angele, Fensel et al. (1998), S. 169.

te Ansatz, der die Wissensakquisition als einen Modellierungsprozess auffasst.[118] Auf dem Modellbasierten Ansatz fußende WBS werden der 2. Generation zugerechnet.[119]

Um Wissensbasierte Systeme Nutzen stiftend anwenden zu können, müssen zuvor einige Anforderungen (implizit) erfüllt sein, die sich auf Art, Verfügbarkeit und den Umfang notwendigen Wissens beziehen.[120] Sowohl implementierungsspezifisches Wissen als auch die Bereitschaft, dieses Wissen zu teilen, muss vorhanden sein; dabei wird auf aufgabenspezifisches Erfahrungswissen zurückgegriffen. Ferner muss eine Explizierbarkeit[121] des Wissens gegeben sein.

2.1.2.4 Ontologien

Im Modellbasierten Ansatz wird eine Ausdifferenzierung von Wissen vorgenommen. Es wird unterschieden zwischen Wissen, das in anderen Bereichen wieder verwendet werden kann, und dem anwendungsspezifischen Instanzenwissen.[122] Für die Repräsentation des wieder verwendbaren Wissens, das repräsentiert in Form eines Modells vorliegt, existieren verschiedene Ansätze. In dieser Arbeit wird auf einen neueren Ansatz, der aus der Forschung zur Künstlichen Intelligenz stammt, zurückgegriffen. Derzeit wird der Ansatz unter den Schlagworten *Ontologien* und *Semantic Web* in einer stetig größer werdenden Forschungsöffentlichkeit diskutiert.[123]

Neben der Verwendung des Begriffs „Ontologie" in der Philosophie[124] findet man in der Neuzeit eine Verwendung des Begriffs in pluralischer Rede vor allem im Bereich des Infor-

118) Siehe hierzu auch die Ausführungen in Kapitel 5.1.2.2, S. 131 ff.

119) Vgl. Puppe, Stoyan et al. (2003), S. 599 f. Bei den Systemen der 2. Generation ist zwischen dem Modell und der Instanziierung des Modells zu unterscheiden. Dies führt zu einer Unterscheidung der Wissensbasis in einem engeren und einem weiteren Sinne. Als eine Wissensbasis im engeren Sinne wird lediglich das repräsentierte Faktenwissen in einem WBS verstanden. Diese Fakten sind die Instanzen des Modells. Bei einer Wissensbasis im weiteren Sinne werden Modell- und Instanzenebene zusammengefasst mit dem Gedanken, dass auch das Modell (bspw. die Ontologie) bereits Wissen enthält. Für die weiteren Ausführungen in der Arbeit wird auf den Zusatz i. e. S. (im engeren Sinn) verzichtet, wenn dieser gemeint wird, und stattdessen lediglich von der Wissensbasis geschrieben. Sollte in Ausnahmefällen die Wissensbasis i. w. S. (im weiteren Sinn) thematisiert werden, so wird dies entsprechend kenntlich gemacht. Siehe hierzu auch das folgende Kapitel.

120) Vgl. VDI/VDE 2621:1996, S. 4.

121) Zum Begriff der Explizierbarkeit siehe auch Fn. 95, S. 24.

122) Der Begriff „Instanz" wurde dem Gebiet der objektorientierten Programmierung entlehnt, wo ein erzeugtes Objekt einer Klasse als Instanz dieser Klasse bezeichnet wird. Einzelne anwendungsspezifische Erfahrungsobjekte, Entitäten, Fakten oder Individuen, die hier allgemein als Instanzen bezeichnet werden, lassen sich zu einem Konzept (in etwa vergleichbar mit einer Klasse in der objektorientierten Programmierung) als Denkeinheit oder Begriff zusammenfassen. Das Wissen über alle Instanzen einer Wissensbasis wird hier auch als Instanzenwissen bezeichnet. Siehe zum weiteren Verständnis von Ontologie und Instanz auch Kapitel 4.1.4.3.2, S. 91 ff., Kapitel 7.2.3.1, S. 223 ff., und Kapitel 7.5, S. 250 ff. Zum Begriff *Konzept* siehe Fn. 141, S. 31.

123) Vgl. Cranefield, Willmott (2002), S. 2; Görz, Wachsmuth (2003), S. 9; Obrst, Liu (2003), S. 123 f.

124) Vgl. bspw. Hartmann (1964); Husserl (1993) oder zur Einführung Jacquette (2002).

mation Systems Research.[125] Hierbei stößt man schnell auf eine in der Hauptsache zweigeteilte Verwendung.[126]

Zum einen dienen Ontologien zur Repräsentation von Wissen. Hierzu gehört bspw. die weithin bekannte Definition von Gruber:[127]

Eine Ontologie ist die explizite Spezifikation einer (geteilten) Konzeptualisierung.

Formal repräsentiertes Wissen basiert auf einer Konzeptualisierung. Für Wissensbasierte Systeme gilt: Was *existiert* ist gleich dem, was repräsentiert werden kann. Das Wissen einer Domäne, repräsentiert in einem Formalismus, entspricht mit seinen Objekten dem *universe of discourse*[128]. Zweck sind das Teilen von Wissen und die Zusammenarbeit von Computerprogrammen basierend auf einer gemeinsam geteilten Konzeptualisierung.[129]

Insbesondere bei einer stark arbeitsteiligen Erfüllung betrieblicher Aufgaben, die auf der Interaktion von Akteursgruppen mit zumindest partiell divergierenden Wissenshintergründen basiert, wird die Bedeutung von Ontologien zur Integration von aufgabenrelevanten Wissenskomponenten deutlich. Ontologien können zur „Integration" aufgabenrelevanter Wissenskomponenten beitragen, indem sie dasjenige sprachlich verfasste Domänenwissen einheitlich strukturieren, das in die Leistungserstellung implizit oder explizit einfließt.[130]

Zum anderen wird der Begriff „Ontologie" einer Spezifikation von Ausdrucksmitteln, unter deren Verwendung eine Domäne beschrieben werden kann, gleichgesetzt. Dies geschieht

---

125) Vgl. die Überblicksbeiträge von Guarino, Giaretta (1995); Guarino (1998); Hesse (2002). Vgl. daneben auch Uschold (1996b), S. 2 ff.; Uschold, Grüninger (1996), S. 96 f.; Gómez-Pérez, Benjamins (1999), S. 1.2.

126) Vgl. Chandrasekaran, Josephson et al. (1999), S. 20. Guarino, Giaretta ermitteln bereits 1995 innerhalb der „Knowledge Sharing Community" die folgenden möglichen Verwendungsweisen für den Begriff Ontologie, die sich jedoch nach Ansicht des Verfassers größtenteils auf die bereits thematisierte Zweiteilung, die im folgenden Fließtext näher erläutert wird, zurückführen lassen oder sich in der Zwischenzeit keiner großen Beliebtheit erfreuen konnten (Ontologie als philosophische Disziplin explizit ausgenommen; vgl. Guarino, Giaretta (1995), S. 1 zu den Verwendungsweisen):
- als philosophische Disziplin,
- als ein informelles konzeptionelles System,
- als eine formale semantische Beschreibung (account),
- als die Spezifizierung einer Konzeptualisierung,
- als eine Repräsentation eines konzeptionellen Systems mittels logischer Theorie entweder charakterisiert durch spezifische formale Eigenschaften oder nur durch den spezifischen Zweck,
- als durch eine logische Theorie genutztes Vokabular und
- als eine (Meta-Level-) Spezifizierung einer logischen Theorie.

Als ein informelles konzeptionelles System liegt die Ontologie „unsichtbar" hinter einer Wissensbasis, d. h. der Wissensbasis liegt eine Ontologie zu Grunde. Als eine formale semantische Beschreibung liegt die Ontologie, die hinter einer Wissensbasis steht, als expliziter formalsprachlicher Ansatz vor (siehe hierzu etwa Guarino, Carrara et al. (1994)). Zur logischen Theorie siehe Fn. 134, S. 30.

127) Gruber (1993), S. 200. Eine ähnlich lautende Definition findet sich bspw. bei Neches, Fikes et al. (1991), S. 40: An ontology defines the basic terms and relations comprising the vocabulary of a topic area as well as the rules for combining terms and relation to define extensions to the vocabulary.

128) Gruber (1993), S. 200.

129) Gruber (1995), S. 907 ff.

130) Alparslan, Dittmann et al. (2002), S. 46.

bspw. bei Wand, Wang (1996)[131], die sich bei ihren Forschungen zur Begriffsmodellierung (*conceptual modeling*) auf ein ontologisches Modell[132] von Bunge stützen.[133] Auch Guarino und Giaretta, die eine Ontologie als eine logische Theorie, die über die beabsichtigte Bedeutung eines formalen Vokabulars, d. h. ihrer ontologischen Verpflichtung gegenüber einer bestimmten Konzeptualisierung der Welt, aufklärt, schließen sich dieser Begriffsverwendung an.[134] Die geplanten Modelle einer logischen Sprache, die ein solches Vokabular verwenden, sind hierbei durch ihre ontologische Verpflichtung (*ontological commitment)* eingeschränkt. Ontologien reflektieren indirekt diese Verpflichtungen und die ihnen zu Grunde liegenden Konzeptualisierungen durch Anpassungen an die geplanten Modelle.[135]

Zusammenfassend findet sich für diese zweite Verwendungsweise des Begriffs „Ontologien" folgende Definition:[136]

Ontologien lassen sich als eine

- explizite und formalsprachliche Spezifikation
- derjenigen „sinnvollen" sprachlichen Ausdrucksmittel auffassen,
- die mehrere Akteure für eine gemeinsam verwendete Konzeptualisierung von realen Phänomenen verwenden,
- die in einem subjekt- und zweckabhängig einzugrenzenden Realitätsausschnitt als wahrnehmbar oder vorstellbar gelten und
- für die Kommunikation zwischen diesen Akteuren benutzt oder benötigt werden.

Die dichotome Verwendung des Begriffs Ontologien fällt bei näherem Hinsehen nicht mehr ganz leicht, weil diese sich gegenseitig stark bedingen: Zelewski stellt fest, dass seine Arbeitsdefinition für Ontologien nicht als eine inhaltliche Abkehr von Gruber, sondern als eine Verschiebung des Schwerpunkts der Akzentuierung auf die sprachliche Dimension aufgefasst

131) Wand selbst behauptet, dass sein Ansatz domänenunabhängig wäre (vgl. Wand (1996), S. 281). Tatsächlich lässt sich Wand´s Ansatz jedoch als Ansatz für eine Commonsense-Domäne begreifen. Zur Unterscheidung von Ontologien nach Arten siehe Kapitel 4.1.2, S. 76.

132) Hier wird der Begriff letztmalig im althergebrachten philosophischen Sinne verwendet. Für den Rest des Textes wird der Begriff der Ontologie im KI-Sinne verwendet, wenn dies nicht anderweitig kenntlich gemacht wurde. Zur besseren Unterscheidung wird die pluralische Rede bevorzugt.

133) Vgl. Wand, Monarchi et al. (1995); Wand (1996); Wand, Wang (1996); Wand, Storey et al. (1999); Parsons, Wand (2000); Wand, Weber (2002).

134) Vgl. Guarino, Giaretta (1995), S. 7: Eine Ontologie ist eine logische Theorie, die eine explizite, partielle Erklärung (account) für eine Konzeptualisierung gibt.

135) Vgl. Guarino (1998), S. 5.

136) Zelewski (2002a), S. 66.

werden soll;[137] Wand stellt fest, dass sein Ansatz tatsächlich als eine Art „*Meta-Ontologie*“ zu der abweichenden ersten Begriffsverwendung angesehen werden kann.[138]

Der Ansatz von Gruber und Kollegen findet weite Verbreitung innerhalb der Forschung zu Ontologien.[139] Um eine Anschlussfähigkeit der Erkenntnisse dieser Arbeit zu gewährleisten, schließt sich der Verfasser dieser vorherrschenden Auffassung an und verwendet im Folgenden die Definition von Gruber.[140] Dabei besteht eine Ontologie aus der Definition von

- *Konzepten*[141],
- *Relationen*[142] und
- *Regeln*

und wird in Wissensbasierten Systemen, die über die Fähigkeit, Schlussfolgerungen zu ziehen, verfügen, eingesetzt.[143]

Weiterführend wird auf Ontologien ab Kapitel 4, Seite 74 ff., eingegangen. Insbesondere wird dabei auf Repräsentationssprachen zur Darstellung von Ontologien eingegangen.

2.1.2.5 Vorgehensmodelle

Vorgehensmodelle werden als allgemeine Anleitungen für die Erfüllung von Aufgaben eingesetzt, um die Planung, Durchführung und Kontrolle von vergleichbaren Problemlösungsprozessen zu erleichtern und die Wiederverwendung von Wissen zu unterstützen. Sie definieren die bei einem Prozess durchzuführenden Aktivitäten, ihre Reihenfolge, die dabei entstehenden Artefakte, die daran beteiligten Personen und weitere benötigte Ressourcen und legen den organisatorischen Rahmen für die Erfüllung von Aufgaben, insbesondere von Projekten, fest.[144]

---

137) Vgl. Zelewski (2005), S. 153.

138) Vgl. Wand (1996), S. 281.

139) Vgl. zur Verwendung der Definition von Gruber bspw. Fensel (2004), S. 3; Grüninger, Lee (2002), S. 39 f.; Lau, Sure (2002), S. 123; Pocsai (2000), S. 39 f. und 77; Staab (2002), S. 200. Zelewski bezeichnet die Ansätze von Gruber und Kollegen als „vorherrschend” (Zelewski (2005), S. 153).

140) Zum Argument der Anschlussfähigkeit vgl. auch Zelewski (2005), S. 144. Zur näheren Untersuchung der dichotomen Verwendungsweise und der Gemeinsamkeiten vgl. Chandrasekaran, Josephson et al. (1999), S. 20 f.

141) Konzepte stellen das Ergebnis eines Konzeptualisierungsprozesses dar. Ein Konzept entspricht der abstrakten Sichtweise auf ein bestimmtes Phänomen eines Realitätsausschnitts in Abhängigkeit von den Erkenntniszielen eines erkennenden Subjekts (vgl. Zelewski (2005), S. 215).

142) Einige Autoren bezeichnen Relationen auch als Attribute, Eigenschaften oder Methoden, vgl. bspw. Erdmann (2001), S. 90 f. Relationen können sowohl einstellig als auch mehrstellig ausgeprägt sein. Einstellige Relationen berücksichtigen als Attribute die Eigenschaften von Konzepten. Mehrstellige Relationen berücksichtigen die Beziehungen von Konzepten untereinander.

143) Die hier kurz vorgestellten Hauptbestandteile einer Ontologie erleichtern wesentlich das Verständnis zu den Ausführungen in den folgenden Kapiteln, weil im Einzelnen immer wieder auf diese konstituierenden Bausteine Bezug genommen wird.

144) Vgl. Boehm (1988), S. 61 und Marciniak, Reifer (1997), S. 597.

Speziell bezieht sich der Verfasser auf Vorgehensmodelle, die die Aktivitäten in den verschiedenen Phasen eines Softwareentwicklungsprojekts über dessen Lebenszyklus widerspiegeln,[145] weil sich aus der Problemstellung der Arbeit ergibt, dass ein systematisches Vorgehensmodell für die Entwicklung eines ontologiebasierten FMEA-Systems benötigt wird.

Folgende Punkte der Nutzenstiftung lassen sich mit dem Einsatz von Vorgehensmodellen für den Anwender anführen[146]:

1. eine Vergleichbarkeit mit erfolgreich abgeschlossenen Projekten aus der Vergangenheit ermöglicht eine erhöhte Planungssicherheit für die Durchführung zukünftiger Projekte,

2. eine effiziente Bearbeitung zukünftiger Projekte durch die Reduzierung von Redundanzen und Nachbearbeitungen wird ermöglicht,

3. eine Reduzierung der Gesamtkosten, die für ein Projekt anfallen, wird durch ein frühzeitiges Reagieren auf Abweichungen im Entwicklungsprozess ermöglicht,

4. durch ein kontrolliertes Vorgehen wird die Qualität der Entwicklungsarbeit sichergestellt,

5. eine Transparenz bei Eigenentwicklungen ermöglicht für Dritte das Nachvollziehen im Projekt gefällter Entscheidungen zu einem späteren Zeitpunkt.

In der Entwicklung softwaretechnischer Systeme lässt sich eine Untermengenbeziehung bezüglich der Anwendung von Vorgehensmodellen vor dem Hintergrund der Problemstellung dieser Arbeit erkennen (siehe auch Abbildung 5, nächste Seite):[147]

- Vorgehensmodelle aus dem Software Engineering[148],
- Vorgehensmodelle aus dem Knowledge Engineering[149] und
- Vorgehensmodelle aus dem Ontology Engineering[150].

145) Vgl. Sommerville (2001), S. 44: Ein Vorgehensmodell ist die abstrakte Repräsentation eines Softwareentwicklungsprozesses.

146) Vgl. Böhm, Wenger (1996), S. 2; Ghezzi, Jazayeri et al. (1991), S. 383; Marciniak, Reifer (1997), S. 597 ff.; Pagel, Six (1994), S. 39 und S. 57 ff.

147) Vgl. zu einer dieser Ausführungen ursprünglichen gesamten Darstellung der drei Bereiche Apke, Dittmann (2003a).

148) Vgl. für einen Überblick über Vorgehensmodelle aus dem Software Engineering Bullinger, Fähnrich (1997), S. 6 ff.; Pomberger, Blaschek (1993), S. 17 ff.; Sommerville (2001), S. 42 ff. und 171 ff.; Wang (2002), S. 280 ff.

149) Vgl. für einen Überblick über Vorgehensmodelle aus dem Knowledge Engineering Haun (2000), S.185 ff., und Beierle, Kern-Isberner (2003), S. 16 ff.

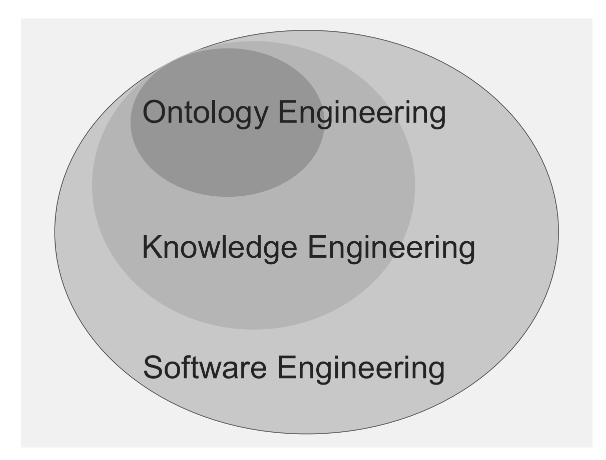


Abbildung 5: Arten von Vorgehensmodellen

Das Software Engineering befasst sich im Allgemeinen mit der Entwicklung von Software auf der Grundlage ingenieursmäßiger Entwicklungsmethoden. Innerhalb der Softwareentwicklung konzentriert sich das Knowledge Engineering auf die Entwicklung Wissensbasierter Systeme[151], d. h. auf Systeme, die zur Zielsetzung haben, Wissen mittels Computer zu verarbeiten. Innerhalb der Entwicklung Wissensbasierter Systeme konzentriert sich wiederum das Ontology Engineering auf die Konzeptualisierung des Wissens einer speziellen Domäne und die formalsprachliche Spezifikation dieser Konzeptualisierung.

150) Vgl. für einen Überblick über Vorgehensmodelle aus dem Ontology Engineering Fernández Lopéz (1999), S. 4.1 ff., und Jones, Bench-Capon et al. (1998), S. 62 ff. Einige Autoren, etwa Holsapple und Joshi (Holsapple, Joshi (2002)), verwenden den Begriff „Ontological Engineering“ statt „Ontology Engineering“; nach Ansicht des Verfassers bezeichnet dieser Begriff in der Übersetzung jedoch eher ein „ontologisches“ Vorgehen, d. h. die explizite Verwendung von wissenschaftlichen Methoden der Philosophie bei der Anwendung ingenieursspezifischer Methoden. Diese Bezeichnung würde den angestrebten Hintergrund dieser Untersuchung jedoch nicht angemessen widerspiegeln, da es sich hier um die Ontologieentwicklung im informationstechnischen Sinne handelt. In der Literatur findet sich zudem auch der hier verwendete Begriff „Ontology Engineering“ etwa bei Kraus (2003), S. 35, und Staab, Studer (2004), S. 115.

151) Siehe hierzu auch Kapitel 2.1.2.3, Seite 24.

2.2 Einordnung der Problemstellung in bestehende Forschungsansätze

In der Literatur lassen sich abgesehen von einer Ausnahme – zu der im nächsten Absatz Stellung genommen wird – keine Ansätze finden, die die Instrumente FMEA und Ontologien als gemeinsamen Forschungsgegenstand haben. Schon gar nicht finden sich Arbeiten, die die drei Instrumente Vorgehensmodell, Ontologien und FMEA gemeinsam untersuchen.

Es konnte lediglich ein Ansatz von Lee ermittelt werden, der die Instrumente FMEA und Ontologien explizit verbindet.[152] Allerdings liegt bei diesem Ansatz das Forschungsinteresse in der Verwendung von FMEAs und Ontologien, um Diagnose-Modelle zu entwickeln. Hierzu wird das Rahmenkonzept DAEDALUS vorgestellt[153] und dargestellt, wie mittels erweiterter FMEAs[154] eine Brücke vom Produktentwicklungswissen zum Diagnosewissen geschlagen werden kann.[155] Die Dependenzen von Ontologien und FMEAs werden nur oberflächlich und ohne wissenschaftlichen Anspruch beschrieben. Es fehlt gänzlich an einer systematischen Vorgehensweise. Erschwerend kommt eine – gegenüber dieser Arbeit –abweichende Verwendung des Instruments *Ontologien* hinzu.[156] Lee's Ansatz leidet unter dem Fehlen einer expliziten Definition von Ontologien und einer fehlenden Repräsentationssprache zur Implementierung von Ontologien. Aus Sicht von Lee scheinen Inferenzen zur Konstitution von Ontologien nicht notwendig zu sein. Für Lee stellt eine Ontologie in etwa ein konzeptuelles Modell (ohne Regeln) dar. Darüber hinaus übergeht er die Darlegung seines Verständnisses einer FMEA; bspw. bleibt er eine Untersuchung der einzelnen Elemente einer FMEA schuldig.[157]

Zum Bereich der gemeinsamen Forschung an Ontologien und Vorgehensmodellen lassen sich in jüngster Zeit einige Ansätze finden. Auf diese Ansätze wird ausführlicher in Kapitel 5.1.2.3, S. 136 ff., eingegangen.

In den folgenden drei Kapiteln werden zunächst die Instrumente *FMEA*, *Ontologien* und *Vorgehensmodelle* eingehender analysiert. Dabei wird der jeweilige State-of-the-art der Forschung isoliert dargelegt. Das heißt, jedes Instrument wird hinsichtlich der aktuellen Forschung unabhängig von den jeweils zwei verbleibenden Instrumenten untersucht. Insbesondere soll bei der Analyse jedes Instruments dessen Einsatzpotential im Umgang mit der Ressource Wissen besonders berücksichtigt werden.

---

152) Vgl. Lee (2001a).

153) Vgl. Lee (2001a), S. 283 ff.

154) Der Autor verwendet hierbei die von ihm in einem vorangegangenen Forschungsprojekt mitentwickelten FMEAs auf Basis Bayesscher Netze (vgl. Lee (2001b)). Diese Modifikation allein bedeutet schon, dass der Forschungsansatz von Lee und der hier vorgestellte Ansatz nicht ähnlich sind.

155) Vgl. Lee (2001a), S. 283.

156) Der Autor versteht unter einer Ontologie ein Konzept-Wörterbuch, das lediglich aus Begriffen und Begriffsrelationen besteht (vgl. Lee (2001a), S. 282 und S. 283). Diese Auffassung entspricht damit allenfalls leichtgewichtigen Ontologien (zur Unterscheidung von leichtgewichtigen und schwergewichtigen Ontologien siehe auch Kapitel 4.1.1, S. 74 f.), die in dieser Arbeit nicht der hier verwendeten Definition von Ontologien zugeschlagen werden, weil Regeln für eine Ontologie als konstituierend angesehen werden (siehe hierzu Kapitel 2.1.2.4, S. 28 ff.).

157) Diese grundlegenden Arbeiten sollen mit dieser Arbeit, neben anderen, nachgeliefert werden.

3 Fehlermöglichkeits- und Einflussanalyse (FMEA)

3.1 Vorstellung des Instruments FMEA

3.1.1 Grundsätze der FMEA

Zur Analyse von Fehlern in Konstruktion und Fertigung wird die Fehlermöglichkeits- und Einflussanalyse (FMEA – auch: Failure Mode and Effects Analysis) herangezogen.

Die FMEA stellt eine formalisierte, analytische Methode[158] dar, deren Ursprünge in den 60er Jahren im Apollo-Projekt der NASA liegen.[159] Die Abbildung 6 enthält einen kurzen Abriss der Entwicklungsgeschichte der FMEA.

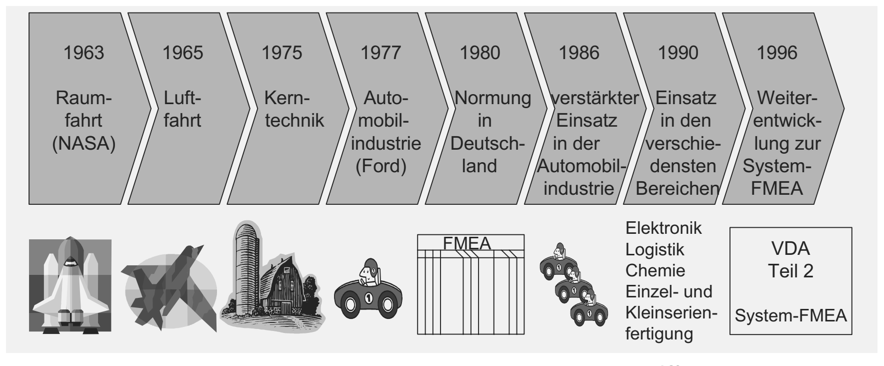


Abbildung 6: Entwicklungsgeschichte der FMEA[160]

Mit Einsatz der FMEA wird eine Vermeidung von Fehlern in einem möglichst frühen Stadium der Entwicklung eines Produkts (d. h. bspw. nach Möglichkeit schon vor dem Beginn der eigentlichen Konstruktion eines Produkts) angestrebt.[161] Insbesondere wird auch der Produktionsprozess in die Betrachtungen einbezogen. Ein Expertenteam identifiziert systematisch mögliche Fehler.[162] Die Ursachen und Auswirkungen werden benannt und mögliche Abstellmaßnahmen aufgezeigt. Bei der vorliegenden induktiven Vorgehensweise wird kein kompletter Ausfall des gesamten Systems angenommen, sondern es werden Fehler in einzelnen Elementen eines Bauteils oder Prozesses angenommen.

Der Grundgedanke der FMEA ist, bereits bei der Planung eines Produkts die Fehler zu eruieren, die eine einwandfreie Funktion behindern können. Die Fehlervermeidung steht bei der

158) Vgl. Stockinger (1989), S. 155.

159) Vgl. DGQ-Band 13-11 (2001), S. 7.

160) Vgl. DGQ-Band 13-11 (2001), S. 8.

161) Von Ahsen und Lange betonen, dass der Einsatz des Instruments (sowie in der Regel aller Qualitätsmanagementinstrumente) mit dem Ziel, eine hohe Kundenzufriedenheit zu erreichen, erfolgt (vgl. von Ahsen, Lange (2004), S. 442).

162) Vgl. DIN 25448:1990, S. 1.

Anwendung einer FMEA im Fokus der Betrachtungen eines Produkts. Gleichzeitig bietet sie sich zum systematischen Auffinden und Beheben von Defiziten im eigentlichen Produktionsprozess an. Qualität soll nicht einfach „erprüft“ werden, d. h. die FMEA soll auch genutzt werden, einen übermäßigen Prüfaufwand aufzudecken und anschließend einzudämmen.[163] Als wesentliche Ziele der FMEA lassen sich nennen:[164]

- kritische Komponenten sollen zur Absicherung der konstruktiven und produktionstechnischen Produktqualität identifiziert werden,
- mögliche Fehler sollen frühzeitig erkannt und lokalisiert werden, um geforderte Funktionen sicherzustellen,
- Risiken sollen abgeschätzt werden,
- Änderungsaufwände nach Beginn der Serienfertigung sollen reduziert werden, d. h. Qualitätsziele eingehalten werden,
- die Anwendung und Weitergabe von Wissen soll ermöglicht und
- die Unterstützung des kontinuierlichen Verbesserungsprozesses im Unternehmen soll erzeugt werden.

Die Vorgehensweise zur Durchführung der FMEA wurde in unterschiedlichen Arbeiten, wie z. B. des VDA, der QS 9000, der DGQ, der SAE (J1739), des DIN und dem MIL-STD 1629A, standardisiert.[165]

Prinzipiell beruht die Vorgehensweise auf den Phasen *Vorbereitung, Suchen, Bewerten* und *Umsetzen.*[166]

Die *Vorbereitung* umfasst die Verankerung des FMEA-Gedankens im Management und die Teambildung im Unternehmen. Nach der organisatorischen Vorbereitung werden die Unter-

163) Vgl. FQS (1994), S. 12.

164) Vgl. Eberhardt (2003), S. 89; Theden (1997), S. 63.

165) Vgl. hierzu beispielhaft VDA-Band 4 (2003), Rabbitt, Bergh (1998), DGQ-Band 13-11 (2001), SAE J1379 (2002), DIN 25448:19990 und MIL-STD 1629A (1980). Der VDA-Band 4 gibt den derzeit gültigen Standard des Verbands der Automobilindustrie wieder. SAE J1379 ist der aktuell gültige Standard eines Bedienungshandbuches zur Anwendung der FMEA. Er ist zurückzuführen auf die Bemühungen der „Großen Drei“ im amerikanischen Automobilbau (DaimlerChrysler, Ford und General Motors). Zu Entwicklungsgeschichte und Inhalten von SAE J1379 vgl. Carlson, McCullen et al. (1996).

166) Zur eigentlichen Durchführung en detail siehe Kapitel 3.1.4.1, S. 46. Nicht zufällig entspricht dieses Meta-Vorgehen dem bekannten Deming-Zyklus, der die Phasen *Plan, Do, Check, Act* (deshalb auch bekannt als PDCA-Zyklus) umfasst (vgl. zu Deming: Zollondz (2002), S. 74 ff. und zum Zyklus insbesondere S. 78 und S. 237 ff.). Unterstellt man, dass alle Qualitätsmanagementkonzepte gemäß PDCA-Zyklus strukturiert sind, so wird erreicht, dass das Instrument FMEA problemlos zur Unterstützung in ein Qualitätsmanagementkonzept integriert werden kann.

suchungsinhalte einer FMEA festgelegt.[167] Der Untersuchungsgegenstand wird bezüglich seines Aufbaus und seiner Funktion strukturiert bzw. beschrieben.[168]

Anschließend wird nach potentiellen Fehlern und deren Folgen *gesucht*.

Das Gefundene wird systematisiert und *bewertet*. Es wird eine Risikoanalyse durchgeführt. Für Fehler, deren Risiken als besonders schwerwiegend für das Unternehmen anzusehen sind, werden Handlungsanweisungen als Maßnahmen zur Fehlervermeidung angesetzt.

Das *Umsetzen* der Maßnahmen zur Fehlervermeidung erfolgt unter expliziter Zuweisung von Verantwortlichkeiten. Im Nachgang erfolgt eine Überprüfung der Wirksamkeit der Maßnahmen zur Fehlervermeidung der durchgeführten Maßnahmen.

Als Hilfsmittel zur Durchführung der FMEA dienen Formblätter. Diese sind oftmals unternehmensspezifisch angepasst. Die Abbildung 7 zeigt ein mögliches Formblatt.[169]

	Fehler-Möglichkeits- und Einflussanalyse ☐ System-FMEA Produkt ☐ System-FMEA Prozess								FMEA-Nr.:
Typ/Modell/Fertigung/Charge:				Sach-Nr: Änderungsstand:		Verantw. Firma:			Abt.: Datum:
System-Nr./Systemelement: Funktion/Aufgabe:				Sach-Nr: Änderungsstand:		Verantw. Firma:			Abt.: Datum:
Mögliche Fehlerfolgen	B	Möglicher Fehler	Mögliche Fehlerursachen	Vermeidungs-maßnahmen	A	Entdeckungs-maßnahmen	E	RPZ	V/T

B: Bewertungszahl für die Bedeutung
V: Verantwortlichkeit
A: Bewertungszahl für die Auftretenswahrscheinlichkeit
T: Termin für die Erledigung
E: Bewertungszahl für die Entdeckungswahrscheinlichkeit
Risikoprioritätszahl RPZ = B*A*E

Abbildung 7: FMEA-Formblatt gemäß VDA'96[170]

167) Eine Reihe von Autoren betont die Wichtigkeit dieser Schritte im Erstellungsprozess einer FMEA, die über die in Kapitel 3.1.4.1, Seite 46, hinausgehen. Vorrangig werden Schritte einer Vorbereitungsphase, die bspw. *Projektorganisation* und *Schulung der Mitwirkenden* (häufig auch *Einführung* oder *Teambildung* genannt) betreffen, genannt (vgl. Dietzsch, Althaus et al. (1999), S. 1395; Metzger (2000), S. 592; Pfeifer (2002), S. 384 ff.; Schiegg, Viertlböck et al. (1999), S. 880; Witter (1995), S. 18 ff.).

168) Vgl. FQS (1994), S. 19.

169) In der Praxis, insbesondere im Bereich des Automobilbaus, werden zurzeit zwei Formblätter bevorzugt: VDA'96 und QS-9000 (vgl. DGQ-Band 13-11 (2001), S. 24). Zur Auswahl eines Formblatts für die weitere Untersuchung siehe auch Kapitel 3.2.3, S. 59 ff.

Im ersten Schritt werden die Systemelemente festgestellt und es wird eine Systemstruktur abgeleitet. Anschließend werden den Systemelementen Funktionen zugeordnet und Funktionsstrukturen aufgezeigt. Im dritten Schritt (Fehleranalyse) werden mögliche Fehlfunktionen festgestellt und es wird eine Fehlfunktionsstruktur erstellt. Bei der darauf folgenden Risikobewertung werden Vermeidungs- und Entdeckungsmaßnahmen erarbeitet. Zusätzlich werden die Bedeutung des Fehlers aus der Sicht des Kunden sowie Auftretens- und Entdeckungswahrscheinlichkeit des Fehlers vor Übergabe an den Kunden festgelegt. In der abschließenden Phase werden Verbesserungsmaßnahmen erarbeitet und es wird eine neue Risikobewertung durchgeführt.

Die in der Risikobewertung ermittelte Risikoprioritätszahl (RPZ), die durch Multiplikation der Faktoren *Bedeutung, Auftreten* und *Entdeckbarkeit* ermittelt wird und Werte von 1 bis 1000 annehmen kann, ist ein Maß für das Gesamtrisiko jedes potentiellen Fehlers. Sie dient als grobes Maß für die Dringlichkeit von Abstellmaßnahmen und als Erfolgsmaßstab für durchgeführte Maßnahmen (Betrag der Differenz zwischen alter und neuer RPZ). Sowohl eine hohe Gesamtzahl als auch ein hoher Wert eines Einzelfaktors sollten hierbei Anlass zum Handeln geben.

3.1.2 Arten der FMEA

Abhängig von Anwendungsfall, Einsatzzeitpunkt und Betrachtungstiefe wird oftmals zwischen System-FMEA (auch Produkt-FMEA), Konstruktions-FMEA (auch Entwicklungs-FMEA) und Prozess-FMEA unterschieden.[171]

3.1.2.1 System-FMEA

Das funktionsgerechte Zusammenwirken der Systemkomponenten in einem (Gesamt-) System und dessen Schnittstellen werden zum Gegenstand der Untersuchungen bei der System-FMEA.[172] Die System-FMEA wird genutzt, um die konstruktiven und produktionstechnischen Aspekte eines Produkts aus ganzheitlicher Sicht zu analysieren.[173] Weil die Betrachtung auf einer höheren Abstraktionsebene als bei der Konstruktions- und Prozess-FMEA erfolgt, sollte die System-FMEA als Basis für die beiden dienen.[174]

---

170) Vgl. VDA-Band 4 (2003), S. 27. Die aktuelle Druckausgabe des VDA datiert aus dem Jahr 2003. Die Vorgabe des aktuell gültigen Formblatts datiert jedoch aus dem Jahr 1996. Es wurden im Jahr 2003 lediglich kleine redaktionelle Änderungen an dem Standard vorgenommen. Wird in der Folge die Quelle zitiert, so wird auf das aktuelle Jahr der Drucklegung Bezug genommen. Wird hingegen der Aufbau des Formblatts als Maßgabe angewendet, so wird auf das Jahr 1996 verwiesen. Der Verfasser befindet sich damit im Einklang mit der üblichen Zitierweise in der Literatur.

171) Vgl. DGQ-Band 13-11 (2001), S. 24 ff.; FQS (1994), S. 15 f.; Witter (1995), S. 10 ff., und Wildemann (1998), S. 12. Bruhn schlägt, analog zur vorgenannten (international) üblichen Systematisierung, eine aufeinander aufbauende Unterteilung in System-, Subsystem- und Prozess-FMEA vor, um Dienstleistungsqualität zu messen (vgl. Bruhn (1997), S. 103).

172) Vgl. DGQ-Band 13-11 (2001), S. 24.

173) Vgl. Nickel (1992), S. 11.

174) Vgl. Witter (1995), S. 10.

3.1.2.2 Konstruktions-FMEA

Zur Analyse von Entwicklungskonzepten wird die Konstruktions-FMEA eingesetzt.[175] Die Konstruktions-FMEA wird parallel zum Entwicklungsprozess durchgeführt.[176] In der Entwicklungsphase gemachte Überlegungen werden in ihr zusammengefasst und können somit der Dokumentation dienen. Unter Berücksichtigung der jeweiligen Funktionserfüllung eines zu konstruierenden Produkts wird hinsichtlich des bestehenden Entwurfs eine Risikoanalyse vorgenommen. Ziel bei der Durchführung der Konstruktions-FMEA ist das Erkennen und Bewerten von Schwachstellen der Entwicklung.

3.1.2.3 Prozess-FMEA

Die Prozess-FMEA wird vor Produktionsbeginn durchgeführt. Nach Möglichkeit baut die Durchführung der Prozess-FMEA auf den Ergebnissen der Konstruktions-FMEA auf.[177] Ausgangspunkt der Betrachtungen zur Erstellung ist in der Regel der geplante Fertigungs- oder Montageprozess für ein bestimmtes Sachgut oder eine bestimmte Dienstleistung. Alle möglichen Störfaktoren und ungeplanten Zustände, die den Ablauf zur Leistungserstellung negativ beeinflussen, sollen aufgezeigt werden.[178] Als Mitwirkende bei der Erstellung einer Prozess-FMEA kommen in der Regel die Mitarbeiter der Arbeitsvorbereitung in Zusammenarbeit mit Mitarbeitern der Qualitätssicherung und der Entwicklung und Konstruktion in Betracht.

3.1.2.4 Zusammenhänge zwischen den FMEA-Arten

Die Tabelle 1 stellt den Zusammenhang zwischen System-, Konstruktions- und Prozess-FMEA exemplarisch bezogen auf die Thematik *Bauwesen* dar. So betrachtet etwa die Konstruktions-FMEA als Betrachtungseinheit eine Fertigteiltreppe mit dem Ziel der Sicherstellung einer anforderungsgerechten Gestaltung und Auslegung.

Als Grundlagen werden die Konstruktionsunterlagen nach Fertigstellung der Konstruktionsplanung herangezogen.

Die Prozess-FMEA nutzt die Erkenntnisse aus der Konstruktions-FMEA, sie kann jedoch auch eigenständig erstellt werden.[179] Bei Nutzung der Erkenntnisse der Konstruktions-FMEA werden deren festgelegte Merkmale bspw. zu Anforderungen, die vom untersuchten Prozess erfüllt werden müssen.[180]

175) Vgl. FQS (1994), S.15.

176) Vgl. Nickel (1992), S. 10.

177) Vgl. FQS (1994), S. 16.

178) Vgl. Witter (1995), S. 11.

179) Vgl. FQS (1994), S. 16.

180) Vgl. DGQ-Band 13-11 (2001), S. 26.

	betrachtetes Objekt	Zielsetzung	Grundlagen der FMEA	Zeitpunkt der Durchführung
System-FMEA	übergeordnetes Produkt / System (z. B. Bauteil)	Sicherstellung einer pflichtenheft-gerechten Funktions-tüchtigkeit, Zuverlässigkeit und Sicherheit	Vorplanung	nach Fertigstellung der Vorplanung
Konstruktions-FMEA	einzelne Bauteile eines Produktes (z. B. Fertigteiltreppe)	Sicherstellung einer pflichten-heft- und fertigungsge-rechten Auslegung	Konstruktions-unterlagen	nach Fertigstellung der Konstruktionsplanung
Prozess-FMEA	einzelne Schritte eines Fertigungs-prozesses (z. B. Betonieren)	Sicherstellung der fehlerfreien Produktion	Bauablaufplanung	nach Fertigstellung der Bauablaufplanung (auch bei bereits laufenden Prozessen)

Tabelle 1: Arten der FMEA und deren Charakteristika

Aufgrund der geschilderten Zusammenhänge unterscheidet Nickel eine duale Vorgehensweise (Konstruktions- und Prozess-FMEA) und eine ganzheitliche Vorgehensweise (System-FMEA) bei der Erstellung einer FMEA. Bei der dualen Vorgehensweise wird ein Produkt in zwei Schritten (auf die Konstruktions-FMEA folgt die Prozess-FMEA) analysiert. Bei der ganzheitlichen Vorgehensweise wird nur eine FMEA in Form einer Produkt-FMEA erstellt.[181] Der deutsche Verband der Automobilindustrie e.V. (VDA) folgt dieser Auffassung noch konsequenter als Nickel in seinen jüngsten Harmonisierungsbestrebungen und fasst System-FMEA und Konstruktions-FMEA zusammen als verschiedene Betrachtungstiefen der *System-FMEA Produkt.* Ferner unterscheidet der VDA neben dieser System-FMEA Produkt noch eine *System-FMEA Prozess*.[182] Der Auffassung des VDA wird sich in der Arbeit angeschlossen. Wo es für die weitere Argumentation unbedeutend ist, wird verkürzend von einer FMEA „gesprochen“.[183]

181) Vgl. Nickel (1992), S. 9 ff.

182) Vgl. VDA-Band 4 (2003), S. 10 ff.

183) Der Verfasser folgt damit bspw. Pfeifer (2001), S. 397. Zu einer erweiterten Begründung zur Verwendung der Vorgabe seitens des VDA siehe auch S. 60.

3.1.3 Abgrenzung von anderen Instrumenten

Neben der FMEA gibt es eine Reihe weiterer Instrumente, um Risikoanalysen aufgaben- und systemspezifisch durchzuführen.[184] Einige artverwandte Beispiele werden nachfolgend kurz beschrieben, um Zusammenhänge darzustellen. Es wird dabei auf eingehendere Erklärungen verzichtet; zur weiteren Vertiefung siehe die Quellen in den Fußnoten.[185]

3.1.3.1 Gefahren-/ Risikoanalyse kritischer Kontroll- bzw. Lenkungspunkte

3.1.3.1.1 Kurzvorstellung des Instruments

Die Gefahren- oder Risikoanalyse kritischer Kontroll- bzw. Lenkungspunkte (auch als Hazard Analysis and Critical Control Point (HACCP) bekannt)[186] wird speziell in der Lebensmittelüberwachung zur Gewährleistung der Lebensmittelsicherheit eingesetzt.[187] Das HACCP-Konzept gilt als ein präventives Instrument zur Gewährleistung der Lebensmittelsicherheit und -qualität. Es behandelt Gefahren, die während des Herstellungsprozesses von Lebensmitteln auftreten und eine Bedrohung für den Verbraucher darstellen können.

Das HACCP-Konzept ist eine systematische Vorgehensweise, die dazu dient, mikrobiologische, chemische und physikalische Fehler schon bei der Lebensmittelherstellung zu identifizieren und durch geeignete Maßnahmen zu beseitigen. Zentraler Bestandteil des Konzepts ist die Festlegung der Critical Control Points (CCPs). CCPs sind zur Überwachung der identifizierten Gefahren nötig. Sie müssen dort eingesetzt werden, wo eine Gefahr eintreten, beseitigt oder vermindert werden kann. Die Anwendung des HACCP-Konzepts führt zu einem implementierten HACCP-System beim einsetzenden Unternehmen.

Im Wesentlichen fußt das HACCP-Konzept auf sieben Grundsätzen:[188]

1. Risikoanalyse und Festlegung von Risikogruppen,
2. Festlegung der CCPs,
3. Festlegung von Grenzwerten für CCPs,
4. Festlegung von Kontrollverfahren für CCPs,

184) Vgl. DGQ-Band 13-11 (2001), S. 74.

185) Der Verfasser hält dieses Vorgehen für gerechtfertigt, weil die ausführliche Darstellung zu weit von der eigentlichen Thematik der Arbeit wegführen würde. Zudem erscheinen die angesprochenen Erkenntniszuwächse dieses Vorgehen zu rechtfertigen, weil die betrachteten weiteren Instrumente in ihrem „Wesen" sehr ähnlich zur FMEA sind und deshalb die Erkenntniszuwächse als „eher gering" erachtet werden.

186) Vgl. NACMCF (1997), S. 1246 ff.

187) Vgl. DGQ-Band 13-11 (2001), S. 74.

188) Vgl. Hulebak, Schlosser (2002), S. 550 f. Für ein spezielles Vorgehen zum HACCP-Konzept basierend auf diesen Grundsätzen aus Sicht des Managements vergleiche Norton (2003).

5. Festlegung von Korrekturmaßnahmen bei Abweichungen von Grenzwerten,
6. Dokumentation des HACCP-Systems,
7. Evaluation des HACCP-Systems.

Das HACCP-Konzept wurde in den 60er Jahren in den USA durch Industrie, Armee und Weltraumbehörde (NASA) entwickelt.[189] Ziel war die garantierte einwandfreie Nahrungsmittelherstellung für Astronauten. Heute gehört das Konzept zum WHO/FAO-Standard[190] als Bestandteil des Codex Alimentarius. Seine Anwendung ist in Europa durch die EG-Richtlinie 93/43 für viele Lebensmittelunternehmen gefordert.[191]

3.1.3.1.2 Diskussion der Unterscheidungsmerkmale

Seitens der HACCP-Richtlinie wird ein Formblatt nicht vorgegeben. Jedoch wird der Anwender in der Regel bei der Durchführung ein Formblatt oder eine Checkliste nutzen, um ein systematisches Vorgehen sicherzustellen.[192] Ähnlich der Durchführung einer FMEA muss eine Priorisierung der kritischen Fehler vorgenommen werden. Eine explizite Erwähnung einer interdisziplinären Zusammenstellung des bearbeitenden Teams findet ebenfalls statt.[193]

Unterschiedlich wird mit ermittelten Fehlern umgegangen. Während bei der Erstellung einer FMEA auch auf die Fehlerfolgen und die Fehlerursachen weiter eingegangen wird, wird bei der Anwendung des Instruments HACCP wesentlich auf die Beobachtung und nach Möglichkeit Beseitigung auftretender Fehler fokussiert.

3.1.3.2 Baumanalysen

3.1.3.2.1 Kurzvorstellung der Instrumente

Aufgrund ihrer besonderen Bedeutung für die Anwendung des Instruments FMEA werden die Fehlerbaumanalyse und die Ereignisablaufanalyse dargestellt und näher erläutert.[194]

---

189) Vgl. Hulebak, Schlosser (2002), S. 549 f.

190) Die Akronyme WHO/FAO stehen für *World Health Organization / Food and Agriculture Organization of the United Nations*.

191) Vgl. Codex Alimentarius (1999), S. 20 ff.; EG-Richtlinie 43 (1993), Artikel 3. Die EG-Richtlinie 93/43 über Lebensmittelhygiene wird oft auch als HACCP-Richtlinie bezeichnet.

192) Für ein Beispiel einer Checkliste siehe Norton (2003), S. 81.

193) Vgl. NACMCF (1997).

194) Für die gemeinschaftliche Anwendung der drei Instrumente FMEA, Fehlerbaumanalyse und Ereignisablaufanalyse siehe Kapitel 3.1.3.2.2, S. 45. Die Fehlerbaumanalyse findet sich für Deutschland genormt in DIN 25424-1:1981.

3.1.3.2.1.1 Fehlerbaumanalyse

Bei der Fehlerbaumanalyse wird ein vorgegebenes unerwünschtes Ereignis auf die Kombination von Primärereignissen zurückgeführt.[195] Das Vorgehen bei der Fehlerbaumanalyse erfolgt *top-down*[196], d. h. ausgehend von einem unerwünschten Ereignis werden mögliche Ausfallkombinationen ermittelt. Die Ausfallkombinationen werden als gerichteter endlicher Graph mit einem Ausgangs- und endlich vielen Eingangsereignissen dargestellt.[197]

Der in der DIN 25424-1:1981 empfohlene Ablauf zur Durchführung einer Fehlerbaumanalyse lässt sich in die drei inhaltlich abgrenzbaren Phasen *Systemanalyse, Erstellung des Fehlerbaums* und *Auswertung des Fehlerbaums* gliedern. Im Einzelnen sind die Schritte gemäß DIN:[198]

1. Durchführung einer detaillierten Untersuchung des Systems mittels einer Systemanalyse. Die Systemanalyse wird wiederum untergliedert in die Untersuchung der *Systemfunktionen*, die Untersuchung der Einwirkungen von *Umgebungsbedingungen*, die Untersuchung der *Hilfsquellen,* die zur Aufrechterhaltung des Systems notwendig sind, die Untergliederung in *Komponenten*, die den technischen Aufbau des Systems bedeuten, und schließlich die Untersuchung der *Reaktion des Systems auf Ausfälle* von Hilfsquellen und Komponenten.

2. Das unerwünschte Ereignis und seine Kriterien, die als Indikator für die Erkennung des Ereignisses dienen, werden festgesetzt. Je nach Aufgabenstellung kann der Ausfall des Systems oder lediglich bestimmter Funktionen des Systems Untersuchungsgegenstand sein.

3. Zuverlässigkeitskenngrößen und zu betrachtende Zeitintervalle für die weitere Analyse werden festgelegt (bspw. die Ausfallhäufigkeit von Komponenten zwischen zwei Inspektionszeitpunkten und die Nichtverfügbarkeit des Systems als Mittelwert der Ausfälle in einem vorgegebenen Zeitintervall).

4. Bestimmung der Ausfallarten der Komponenten. Die Ergebnisse aus Schritt 1 zur Untersuchung der Reaktion des Systems auf Ausfälle von Hilfsquellen und Komponenten werden hierzu herangezogen.

---

195) Vgl. DIN 25419:1985, S. 1.

196) In den Normen DIN 25419:1985 und DIN 25424-1:1981 wird jeweils von einem *induktiven* bzw. *deduktiven* Vorgehen gesprochen; um Verwechslungen mit dem Begriffsverständnis aus der formalen Logik, die an anderer Stelle verwendet wird, zu vermeiden (hinsichtlich der wahrheitserhaltenden Stringenz von geschlussfolgerten Aussagen), werden im folgenden Text die Begriffe *bottom-up* (für induktives Vorgehen) und *top-down* (für deduktives Vorgehen) zur Charakterisierung von Vorgehensweisen verwendet.

197) Diese Darstellung gilt als Namensgeber des Instruments *Fehlerbaum*analyse.

198) Vgl. DIN 25424-1:1981, S. 4. Für eine genauere Erklärung zu den einzelnen Schritten siehe ebenda oder Pfeifer (2002), S. 337 ff. Schritt 1 im Text entspricht der vorgenannten Systemanalyse, die Schritte 2 bis 6 der Erstellung des Fehlerbaums und die Schritte 7 und 8 entsprechen der Auswertung des Fehlerbaums.

5. Aufstellen des Fehlerbaums.[199] Bei der Aufstellung werden Ausfälle kategorisiert.[200]

6. Kenngrößen, wie Ausfallhäufigkeiten und Nichtverfügbarkeiten, werden als Eingänge in den Fehlerbaum zusammengestellt.

7. Der Fehlerbaum wird ausgewertet, bspw. werden Eintrittshäufigkeiten von Ausfallkombinationen und unerwünschten Ereignissen ermittelt. Je nach Zielsetzung kann eine quantitative oder qualitative Auswertung erfolgen.[201]

8. Die Ergebnisse der Auswertung werden bewertet.

Als wesentliche Ergebnisse der Fehlerbaumanalyse lassen sich festhalten:[202]

1. Es liegt eine systematische Erfassung von zum unerwünschten Ereignis führenden Ausfallkombinationen vor.

2. Eintrittshäufigkeiten der Ausfallkombinationen werden ermittelt.

3. Eintrittshäufigkeiten des unerwünschten Ereignisses werden bestimmt.

4. Kleinste Ausfallkombinationen werden ermittelt, die zum unerwünschten Ereignis führen.

3.1.3.2.1.2 Ereignisablaufanalyse

In der Ereignisablaufanalyse[203] werden Ereignisse ermittelt, die sich aus einem vorgegebenen Anfangsereignis entwickeln können.[204] Das Instrument wird eingesetzt, um die Abläufe von Ereignissen zu beschreiben und zu bewerten. Bevorzugt wird die Ereignisablaufanalyse bei der Untersuchung von technischen Systemen in Bezug auf potentielle Störfälle eingesetzt.[205]

Ausgehend von einem Anfangsereignis (bspw. Ausfall einer Pumpe) werden Folgeereignisse (Leistungsreduzierung im Kühlkreislauf) bis zu möglichen Endzuständen (Überschreitung zulässiger Hüllrohrtemperatur), die selbst ein Ereignis darstellen, ermittelt. Zur Durchführung der Ereignisablaufanalyse werden den Systemzustand beeinflussende Ereignisse beschrieben

---

199) Die Aufstellung des Fehlerbaums folgt einem festgelegten Vorgehen, das mittels eines Ablaufdiagramms von Pfeifer nachvollziehbar dargestellt und erläutert wird. Aus Platzgründen sei an dieser Stelle hierauf verwiesen: Pfeifer (2002), S. 340 f.

200 Vgl. hierzu DIN 25424-1:1981, S. 5.

201) Vgl. hierzu weiterführend DIN 25424-1:1981, S. 6 und Pfeifer (2002), S. 343.

202) Vgl. DIN 25424-1:1981, S. 6.

203) Die Ereignisablaufanalyse findet sich für Deutschland genormt in DIN 25419:1985. Die Ereignisablaufanalyse wird als Ergebnis der Übersetzung aus dem Englischen (Event Tree Analysis (ETA)) auch als *Ereignisbaumanalyse* bezeichnet, vereinzelt findet sich auch die Namensgebung *Störfall*ablaufanalyse (vgl. DIN 25424-1:1981, S. 1). In DGQ-Band 13-11 (2001) wird der Fehler begangen, Ereignisbaumanalyse und Störfallablaufanalyse als zwei unterschiedliche Instrumente zu betrachten (S. 74 und 76).

204) DIN 25419:1985, S. 1.

205) Vgl. DIN 25419:1985, S. 1.

und in eine kausale oder zeitliche Beziehung zueinander gebracht und in einem Diagramm abgebildet.[206] Das entwickelte „logische Modell“ gibt die Bedingungen an, unter denen ein Anfangsereignis durch bestimmte Folgeereignisse zu einem bestimmten Endzustand gelangt.[207] Mittels Wahrscheinlichkeiten, die als Schätzwerte für das Auftreten der Ereignisse angesetzt werden, kann eine Auswertung des Diagramms erfolgen.

3.1.3.2.2 Diskussion der Unterscheidungsmerkmale

Für Schritt 4 der Fehlerbaumanalyse wird in der DIN 25424-1:1981 explizit auf die FMEA als Hilfsmittel zur erleichterten Erstellung des Fehlerbaums hingewiesen.[208]

Die FMEA geht im Gegensatz zur Fehlerbaumanalyse, die Ausfallkombinationen betrachtet, lediglich von einzelnen Ausfällen aus. Deshalb sind Fehler, Fehlerursache und Fehlerfolge mit einem „exklusiven Oder“ verbunden. Es wird jeweils nur ein Fehlertupel betrachtet.

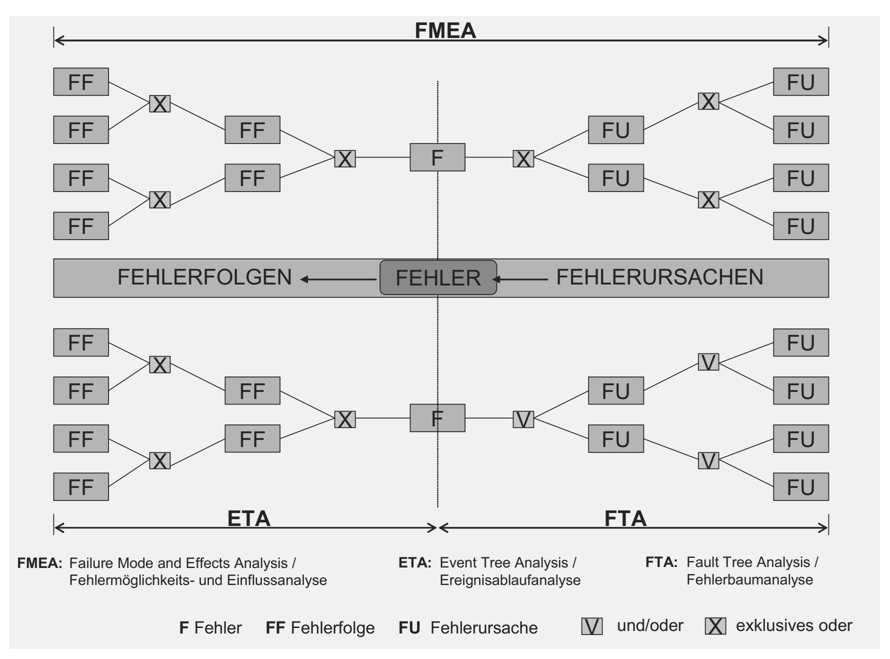


Abbildung 8: Zusammenhänge von FMEA, FTA und ETA[209]

206) Vgl. DGQ-Band 13-11 (2001), S. 74. Die einschließende ODER-Verknüpfung, die NICHT-Verknüpfung und die UND-Verknüpfung treten in der Ereignisablaufanalyse nicht auf (vgl. DIN 25419:1985, S. 1). Lediglich die ausschließende ODER-Verknüpfung findet Verwendung.

207) Vgl. DIN 25419:1985, S. 2. Graphische Symbole für die Darstellung der betrachteten Abläufe finden sich ebenda.

208) Vgl. DIN 25424-1:1981, S. 5.

209) In Anlehnung an DGQ-Band 13-11 (2001), S. 75.

Von der Ereignisablaufanalyse unterscheidet sich die Fehlerbaumanalyse, indem bei der ersten die unerwünschten Ereignisse, resultierend aus bestimmten Ursachen, gesucht werden. Hingegen gibt man bei der Fehlerbaumanalyse das unerwünschte Ereignis vor und sucht anschließend nach den Ursachen.[210] Betrachtet man bei der FMEA nur das „Kern“-Vorgehen, so lässt sich von einem Bottom-up-Vorgehen sprechen.[211] Ausgehend von einem Fehler werden dessen Auswirkungen und anschließend die möglichen Abstellmaßnahmen hierzu untersucht.

Abbildung 8 verdeutlicht die Betrachtungszusammenhänge von FMEA, FTA und ETA. Im oberen Bereich findet sich eine abstrakte Darstellung einer FMEA-Argumentkette[212] und im unteren Bereich die Argumentketten von FTA und ETA mit dem gemeinsamen Ausgangspunkt *Fehler*.

3.1.4 Anwendung der FMEA

In dieser Arbeit wird das Instrument FMEA insbesondere in Hinsicht auf seine Verwendbarkeit zur Entwicklung und Verwendung von Ontologien analysiert. Deshalb steht der Erstellungsprozess im engeren Sinne als „Durchführung“ im Fokus der Untersuchung. Auf Aspekte wie *Projektorganisation* und *Einführung in die Erstellungsweise einer FMEA* wird nur eingegangen, sofern die Aspekte für die Weiterverwendung in der Arbeit notwendig sind.

3.1.4.1 Durchführung einer FMEA

Die Durchführung einer FMEA erfolgt in fünf Schritten (siehe auch Abbildung 9): Der *Strukturanalyse* folgt die *Funktionsanalyse,* dem schließen sich die *Fehleranalyse* und die *Risikobewertung* an. Abschließend folgt die *Optimierung.*[213]

210) Vgl. DIN 25424-1:1981, S. 1.

211) Vgl. Kato, Shirakawa et al. (2002), S. 1.

212) Siehe zum Verhältnis von *Fehler*, *Fehlerfolge* und *Fehlerursache* auch Kapitel 3.3.4, S. 67 f. In diesem Kapitel wird die Abhängigkeit dieses Tripels von der Betrachtungsebene deutlich.

213) Vgl. DGQ-Band 13-11 (2001), S. 18 ff. Diese Strukturierung entspricht der von Seite 36 im engeren Sinne. So wird bei der hier vorliegenden Strukturierung im engeren Sinne auf die Phase *Vorbereitung*, die bspw. die Einführung in die Erstellungsweise einer FMEA und die Projektorganisation beinhalten würde, verzichtet. Die vorliegenden fünf Schritte einer Erstellung einer FMEA im engeren Sinne lassen sich den „Meta“-Phasen *Suchen* (Strukturanalyse, Funktionsanalyse, Fehleranalyse), *Bewerten* (Risikobewertung) und *Umsetzen* (Optimierung) zuordnen.
Der Begriff „Optimierung“ wird als Eigenname aus der gängigen (standardisierten) FMEA-Literatur übernommen, auch wenn in der Betriebswirtschaftslehre eine abweichende Verwendung angetroffen wird, um die Anschlussfähigkeit der Erläuterungen an die FMEA-Literatur zu gewährleisten.

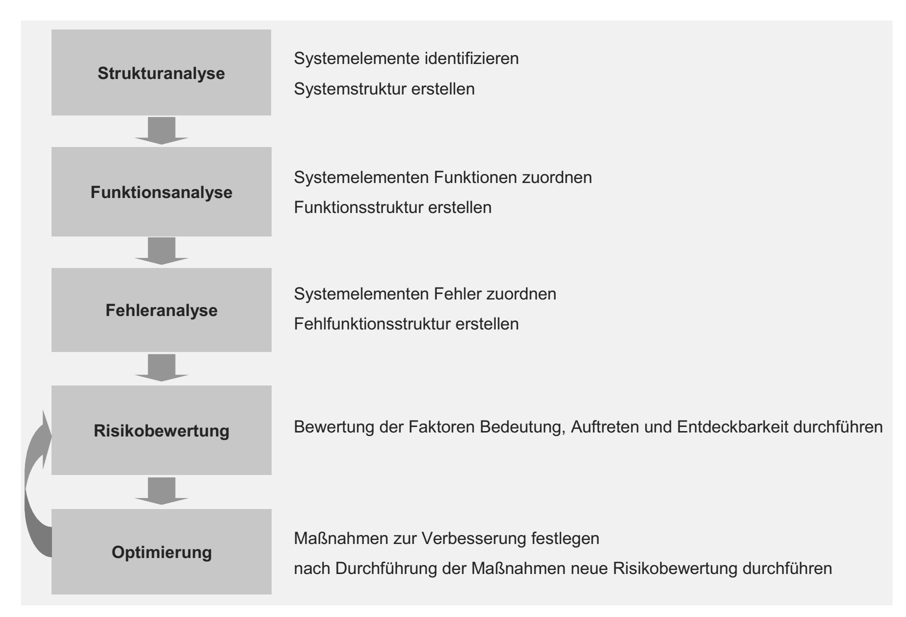


Abbildung 9: Fünf Schritte zur Durchführung einer FMEA[214]

3.1.4.1.1 Schritt *Strukturanalyse*

Der Untersuchungsgegenstand wird üblicherweise als System verstanden, das aus einzelnen Systemelementen besteht (siehe Abbildung 10). Im ersten Schritt *Struturanalyse* werden die Systemelemente eines Systems identifiziert und zu einer hierarchischen Struktur angeordnet. Das zu untersuchende System bedarf einer genauen Definition (falls notwendig, auch Einschränkungen der Systemgrenzen) hinsichtlich seines zu berücksichtigenden Umfangs. Dabei definiert die Festlegung der Systemgrenzen das untersuchte System.[215] Von vornherein ist die Granularität, d. h. die Betrachtungstiefe, nicht reglementiert, sondern steht in Abhängigkeit zum Untersuchungsgegenstand, d. h., es wird weiter untergliedert, so lange die Mitwirkenden des FMEA-Teams dies bei der Erstellung für notwendig erachten. In einer Top-down-Vorgehensweise werden die einzelnen Systemelemente untergliedert und die Schnittstellen zwischen den Systemelementen werden ermittelt.[216] Eine Schnittstelle bei der System-FMEA Produkt existiert oftmals an vorhandenen physikalischen Verbindungen von Systemelementen

214) In Anlehnung an VDA-Band 4 (2003), S. 15.

215) Vgl. Eberhardt (2003), S. 89.

216) Vgl. Schloske (1999), S. 31. Als Systemelemente lassen sich bspw. Baugruppen, Bauteile oder Prozessschritte unterscheiden.

(bspw. sich berührende Systemelemente (Schraube/Mutter) und an zerlegten Systemelementen durch willkürliche Festlegung der Systemgrenzen (Kabelverbindungen)).[217]

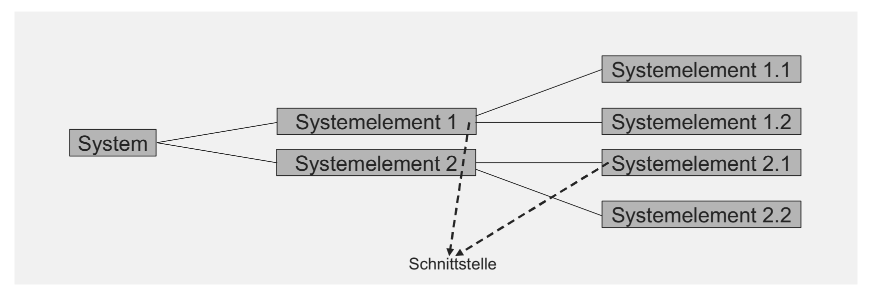


Abbildung 10: FMEA-Systemstruktur mit Schnittstelle

Im Allgemeinen wird die entwickelte Systemstruktur auch als System- oder Strukturbaum bezeichnet.[218] Dabei muss sichergestellt werden, dass jedes Systemelement nur einmal existiert, damit eine eindeutige Abbildung des Systems ermöglicht wird.[219]

3.1.4.1.2 Schritt *Funktionsanalyse*

Den ermittelten Systemelementen werden im nächsten Schritt (der Funktionsanalyse) Funktionen zugeordnet. Eine Funktion entspricht einem Merkmal, das einem Systemelement eine Aufgabe oder Wirkung zuweist. Die Ermittlung möglicher Funktionen und ihre Zuordnung zu einem Systemelement werden durch die Funktionsanalyse abgedeckt. Die Funktionsanalyse führt als Ergebnis zu einer Funktionsstruktur. Die Funktionen der Systemelemente werden miteinander verknüpft und in einer Funktionsstruktur dargestellt (Funktionszuordnung). Dabei werden sich ergebende Teilfunktionen untereinander „logisch" verknüpft, so dass sie in ihrer Summe eine Funktion darstellen.[220] Die Funktionsstruktur beschreibt die funktionalen Zusammenhänge zwischen den Systemelementen.[221]

---

217) Vgl. VDA-Band 4 (2003), S. 16. Siehe hierzu auch die Abbildung 10. Bspw. könnte es sich bei Systemelement 1 um ein Bedienpult handeln und bei Systemelement 2.1 um eine Schraubmaschine, die mit einem Datenkabel zwecks Steuerung miteinander verbunden sind.

218) Vgl. Schloske (1999), S. 31.

219) Vgl. VDA-Band 4 (2003), S. 16.

220) Bspw. lässt sich die Funktion „Wasser abpumpen" in die Funktionen „Zylinder mit Wasser füllen" und „Zylinder mit Wasser leeren" gliedern. Beide Teilfunktionen ergeben „logisch" die Funktion „Wasser pumpen".

221) Für ein Konzept einer rechnergestützten semantischen Funktionsstrukturmodellierung vgl. Puri (2003), S. 76 ff. Auf Seite 93 derselben Quelle findet sich ein Aktivitätsdiagramm zur Erstellung einer Funktionsstruktur.

3.1.4.1.3 Schritt *Fehleranalyse*

Basierend auf System- und Funktionsstruktur wird im dritten Schritt die Fehleranalyse vollzogen.[222] Es wird vom untersuchten Systemelement ausgegangen und eine Fehleranalyse aufgebaut, die *Fehler (F)* des betrachteten Systemelements, *Fehlerursachen (FU)* und *Fehlerfolgen (FF)* einschließt.[223]

Arbeitsgrundlage für die Fehleranalyse ist zumeist ein Formblatt.[224]

3.1.4.1.3.1 Teilschritt *Fehler*

Aufbauend auf den ermittelten Funktionsstrukturen aus der Funktionsanalyse werden Fehlfunktionen (mögliche Fehler) abgeleitet. Mehrere Fehlermöglichkeiten können dabei einer Funktion zugeordnet werden.[225]

Gemäß der Fehlerdefinition in Kapitel 2.1.1.3, S. 20, wird dabei angenommen, dass die vom Systemelement (SE) geforderte Funktion als Anforderung teilweise oder gänzlich nicht erfüllt wird. Ferner wird angenommen, dass das Auftreten eines Fehlers möglich, jedoch nicht notwendig sein muss.[226] Häufig erfolgt die Ermittlung von Fehlfunktionen durch eine einfache Negation der Erfüllung der Funktionen jener Elemente der Funktionsstruktur.[227]

Die Fehlfunktionen werden als Fehlfunktionsstrukturen abgebildet (Abbildung 11).

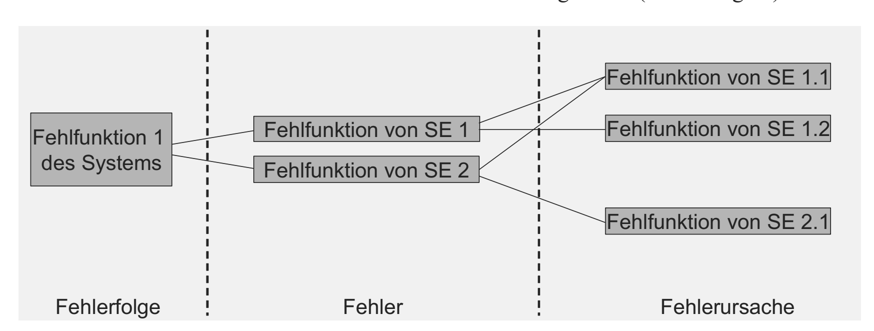


Abbildung 11: Fehlfunktionsstruktur

222) In Bezug auf die Aufgabenstellung kann die Fehleranalyse mit Diagnoseproblemen verglichen werden (vgl. Nickel (1992), S. 42 ff.).

223) Vgl. VDA-Band 4 (2003), S. 12.

224) Vgl. Witter (1995), S. 22.

225) Vgl. DGQ-Band 13-11 (2001), S. 29.

226) Vgl. Schloske (1999), S. 36. Die Annahme aller möglichen Fehler soll sicherstellen, dass alle in Betracht kommenden Fehler Berücksichtigung finden bei der Analyse.

227) Dieses Vorgehen ähnelt dem Vorgehen bei der Fehlerbaumanalyse, siehe Kapitel 3.1.3.2.1.1, S. 43.

3.1.4.1.3.2 Teilschritt *Fehlerursache*

Den ermittelten Fehlern werden Ursachen für ihre Entstehung zugeordnet. Alle potentiellen Fehlfunktionen der untergeordneten Systemelemente und der über Schnittstellen zugeordneten Systemelemente werden als mögliche Fehlerursachen betrachtet (z. B. das Versagen eines elektrischen Sicherheitsschalters).[228]

3.1.4.1.3.3 Teilschritt *Fehlerfolge*

Gemäß SAE J1379 wird die Fehlerfolge definiert als Auswirkung einer Fehlfunktion, wie sie vom Kunden wahrgenommen wird.[229] Den ermittelten Fehlern werden mögliche Folgen zugeordnet.[230] Hierzu müssen diese möglichen Folgen zunächst ebenfalls ermittelt werden. Um mögliche Fehlerfolgen aufzufinden, empfiehlt Stamatis, einige Dokumente hinsichtlich möglicher Informationen zu untersuchen:[231]

- Garantie-/Gewährleistungen,
- Kundenbeschwerden,
- Informationen des Außendiensts,
- Verlässlichkeitsdaten und
- Machbarkeitsstudien.

Als Ergebnis erhält man nach der Durchführung der Fehleranalyse eine vollständige Fehlfunktionsstruktur, die anschließend direkt in ein Formblatt übertragen werden kann, sofern sie nicht schon direkt dort erstellt wurde.

3.1.4.1.4 Schritt *Risikobewertung*

Für die Fehlerursachen werden jeweils Risikoprioritätszahlen (RPZ) ermittelt. Mittels Multiplikation ergibt sich die RPZ aus:[232]

---

228 Vgl. Schloske (1999), S. 37.

229) Vgl. SAE J1379:2002, S. 10. Hierbei ist jedoch zu berücksichtigen, dass die Menge der Kunden im Sinne eines umfassenden Qualitätsmanagements (TQM) nicht nur den „eigentlichen“ Endkunden, sondern bspw. auch den abnehmenden verantwortlichen Mitarbeiter des anschließenden Produktionsschritts umfassen kann.

230) Die Fehlerfolgen lassen sich gemäß Schloske in zwei Arten gliedern: Fehlerfolgen auf das Produkt und Fehlerfolgen auf den Herstellungsprozess bezogen (vgl. Schloske (1999), S. 38). Schloske weist darauf hin, dass die Untersuchung der Fehlerfolgen auf den Herstellungsprozess bezogen im Allgemeinen bisher bei der Anwendung des Instruments FMEA vernachlässigt wurde, aufgrund der Bedeutung für die Rentabilität (z. B. durch Verringerung von Stillstandszeiten der Produktionseinrichtung) jedoch von Interesse für das Unternehmen sein sollten.

231) Vgl. Stamatis (2003), S. 115.

232) In der Literatur finden sich auch namentlich abweichende Dreigliederungen für die Ermittlung der RPZ mittels Multiplikation, so z. B. in Witter (1995), S. 23: *Schwere der Fehlerauswirkung* entspricht *S*.

1. B: Bewertungszahl für die Bedeutung,
2. A: Bewertungszahl für die Auftretenswahrscheinlichkeit,
3. E: Bewertungszahl für die Entdeckungswahrscheinlichkeit.

Als Basis für die Risikobewertung werden die bereits bekannten Maßnahmen zur Vermeidung und Entdeckung von Fehlern im Formblatt hinterlegt. Die Bewertungszahl B wird für die Bedeutung einer Fehlerfolge aus Kundensicht festgelegt. Für die Bewertungszahl A, die die Auftretenswahrscheinlichkeit jeder Fehlerursache unter Berücksichtigung aller zum Zeitpunkt der Durchführung wirksamen Vermeidungsmaßnahmen darstellt, wird für jede Fehlerursache ein Wert festgelegt. Alle Maßnahmen, die zur Entdeckung von Fehler, Fehlerfolge und Fehlerursache dienen, werden als Entdeckungsmaßnahmen im Formblatt hinterlegt. Es wird eine Bewertungszahl für die Entdeckungswahrscheinlichkeit jeder Fehlerursache unter Berücksichtigung der zum Zeitpunkt der Erstellung wirksamen Entdeckungsmaßnahmen ermittelt.[233] Für B-Bewertung, A-Bewertung und E-Bewertung werden in der Regel Bewertungszahlen von jeweils 1 bis 10 verwendet.[234] Mittels eines Ergebnisvergleichs lässt sich das Verhältnis der einzelnen Fehlerursachen anhand der ermittelten RPZ zueinander darstellen. Fehler, Fehlerfolgen und Fehlerursachen mit den höchsten Risikoprioritätszahlen oder den schwerwiegendsten einzelnen Bedeutungen sollten vorrangig vermieden werden.[235]

3.1.4.1.5 Schritt *Optimierung*

Die Basis des Schritts *Optimierung* stellen die Ergebnisse der Risikobewertung dar. Diese verdeutlichen Risiken, die als Ansatzpunkte für Verbesserungs- und Vermeidungsmaßnahmen dienen können. Dabei weist eine hohe RPZ oder ein hoher Einzelwert von B-Bewertung, A-Bewertung oder E-Bewertung auf eine notwendige Maßnahmenergreifung hin.

Nach der Durchführung von Verbesserungs- und Vermeidungsmaßnahmen während des Schritts *Optimierung* wird eine erneute Risikobewertung durchgeführt.

---

233) Die Bewertung der Entdeckungswahrscheinlichkeit der aufgetretenen Fehlerursache wird unter Berücksichtigung aller dafür aufgelisteten Entdeckungsmaßnahmen durchgeführt (VDA-Band 4 (2003), S. 21). Weil sich aufgelistete Entdeckungsmaßnahmen auch auf Fehler und Fehlerfolgen beziehen können, kommt es zu einer „gemeinsamen E-Bewertung“ (VDA-Band 4 (2003), S. 21), d. h. die Entdeckungswahrscheinlichkeit einer Fehlerursache resultiert aus einer Kombination von Fehler, Fehlerursache und Fehlerfolge. Ähnlich argumentiert auch die DGQ, indem sie empfiehlt, die Entdeckungsmaßnahme vorzugsweise auf die Fehlerursache zu beziehen und gleichzeitig einräumt, dass aus technischen Gründen oder Kostengründen auch eine Entdeckung des Fehlers oder der Fehlerfolge angebracht sein kann (vgl. DGQ-Band 13-11 (2001), S. 19).

234) An dieser Stelle wird auf die Vorstellung einer Skala verzichtet, weil diese nicht von Relevanz für die weitere Argumentation in dieser Arbeit ist. In der Literatur finden sich jedoch zahlreiche Beispiele für Arten von Bewertungsskalen: vgl. SAE J1739:2002, S. 11 und S. 13; Stamatis (2003), S. 117; VDA-Band 4 (2003), S. 22 f.; Wildemann (1998), S. 35 ff.

235) Eine feststehende Eingriffsgrenze in Form eines Vorgabewerts für die RPZ gilt allgemein als nicht sinnvoll, weil die Maßstäbe der Bewertung stark differieren können (vgl. DGQ-Band 13-11 (2001), S. 19; VDA-Band 4 (2003), S. 24; Witter (1995), S. 28).

3.1.4.2 Einsatzgebiete für die FMEA

In den letzten Jahren lässt sich eine kaum mehr überschaubare Weiterentwicklung des Instruments FMEA in zahlreichen Einsatzgebieten in der Literatur finden. Deshalb wird an dieser Stelle – mittels einer einfachen Systematik – versucht, den Stand der Forschung zu gliedern (siehe Abbildung 12).[236]

Ein Großteil der Forschungsmühen zum Themenbereich FMEA bezieht sich auf eine Modifikation der Untersuchungs*systematik* einer FMEA, d. h. eine Änderung des Konzepts wird angestrebt. Neben der Modifikation der Untersuchungssystematik, die oftmals eine Erleichterung der Durchführung einer FMEA durch entsprechende Hilfsmittel anstrebt, kann ferner bei den Forschungsmühen eine Modifikation des Untersuchungs*objekts* bei der Durchführung einer FMEA festgestellt werden, d. h. die „ursprünglichen" Einsatzgebiete für die Anwendung einer FMEA werden verlassen und „neue" Einsatzgebiete werden erschlossen.[237]

Es lässt sich weiterhin in *IT-basierte*, d. h. durch den Einsatz von Informationstechnik gestützte, und zum anderen in *Formblatt-basierte* FMEA-Anwendungen, d. h. „ursprüngliche" Anwendungen, wie sie mit der anfänglichen Entwicklung der FMEA beabsichtigt wurden, unterscheiden.

Diese vier Entwicklungstendenzen werden in Abbildung 12 synoptisch dargestellt.[238]

IT-basiert	**IT- basierte Änderung der Vorgehensweise**	**IT-basierter Einsatz in weiteren Anwendungsgebieten (Fokus)**
Formblatt-basiert	**Änderung der herkömmlichen Vorgehensweise**	**Einsatz in weiteren Anwendungsgebieten (Fokus)**
	Modifikation Untersuchungssystematik	Modifikation Untersuchungsobjekt

Abbildung 12: Entwicklungstendenzen der FMEA

236) Die einfache Systematik dient an dieser Stelle lediglich der Verdeutlichung. Bei den zu nennenden Forschungsansätzen kann nicht immer eine eindeutige Zuordnung vorgenommen werden, weil die Forschungsansätze in Teilen auch die übrigen Bereiche der Systematik betreffen können. Für den Fall einer solchen Überschneidung wird der jeweilige Ansatz gemäß seinem identifizierten Schwerpunkt in die Systematik eingeordnet.

237) Siehe hierzu auch die Abbildung 6, S. 35. Als „ursprüngliche" Einsatzgebiete für Anwendungen der FMEA lassen sich hier zusammenfassend die Raumfahrt-, die Luftfahrt- und die Automobilindustrie nennen.

238) Eine Unterscheidung hinsichtlich der Entwicklungstendenzen zur Verbesserung der FMEA in Konzeptänderung und Softwareunterstützung findet sich bspw. auch in Ruta (1999), S. 57.

Entwicklungen *Formblatt-basierter Ansätze zur Modifikation der Untersuchungssyste-matik* konzentrieren sich vornehmlich auf Veränderungen am Layout des Formblatts oder auf die Erleichterung und intersubjektive Nachvollziehbarkeit der Ermittlung von Risikoprioritätszahlen. Ausführungen hierzu finden sich bspw. in: Schiegg, Viertlböck et al. (1999) und von Ahsen, Lange (2004) zur Weiterentwicklung der RPZ-Ermittlung und in Schmidt, Minges et al. (1991) zum Layout. Weitere Beiträge zu Modifikationen der Formblatt-basierten Untersuchungssystematik liefern für den Bereich der Einbindung von Zulieferern Edenhofer, Baumann et al. (2002) und Pfeifer, Reinecke et al. (2000); für den Bereich der Integration weiterer Techniken (bspw. Kreativitäts- und Analysetechniken) Gimpel, Stolten et al. (2002); Herb, Herb et al. (1998); Tammler, Eschborn (1998) und, für den Übergang von Listen zu Matrizen, Metzger (2000).

Forschungsansätze zur *Formblatt-basierten Erstellung von FMEAs mit einem modifizierten Untersuchungsobjekt* streben eine Erweiterung des Anwendungshorizonts des Instruments FMEA an. So findet man Ansätze, FMEAs nach besonderen Untersuchungsobjekten zu systematisieren. Zum einen wird der Fokus auf ein Untersuchungssobjekt vergrößert. So konzentriert sich die Human-FMEA von Algedri, Frieling et al. (2000) bspw. zusätzlich besonders auf menschliche tätigkeitsorientierte Handlungsfehler. Die Notfall-FMEA von Pieper (1998) soll die Erstellung eines Referenzhandbuchs für Notfälle ermöglichen und erweitert das herkömmliche Untersuchungsobjekt um logistische Aspekte. Zum anderen übertragen Forschungsansätze den Einsatzfokus der FMEA auf „gänzlich neue“ Anwendungsgebiete, so bspw. Mäckel (2001) auf die Software-Entwicklung; Deutsch, Mücke (1999) auf Verfügbarkeiten von Sondermaschinen und Eberhardt (2003) auf die Gefährdungsanalyse gemäß der EU-Richtlinie für Maschinen.

Die derzeitigen IT-basierten Modifikationen der Untersuchungssystematik einer FMEA-Durchführung kann nach drei Kategorien unterschieden werden:[239]

- Anwendungssoftware ausschließlich zum Editieren von Formblättern
  Siehe Seite 54 f.

- Anwendungssoftware als eigenständige spezialisierte FMEA-Module
  Hierzu beschreiben Johne, Zieglowski (2000), Sponner, Hoffmann et al. (2000), Vollrath (2000) jeweils eine datenbankbasierte Software. Ansätze zur Entwicklung von Wissensbasierten Systemen finden sich in FQS (1994); Nickel (1992); Struss (2004) und Wirth, Berthold et al. (1996).

- Anwendungssoftware im Rahmen eines CAQ-Systems
  Pfeifer, Reinecke et al. (1998) stellen einen Ansatz für ein verteiltes System vor, in dem FMEA-Daten mittels SGML ausgetauscht und anderen Teilsystemen zur Verfügung gestellt werden.

Vornehmlich wird bei IT-basierten Modifikationen der Untersuchungssystematik eine Wiederverwendung von Wissen in Form bereits erstellter FMEAs angestrebt, so bspw. bei Strunk

239) Vgl. DGQ-Band 13-11 (2001), S. 68.

(2000). Begleitet werden diese Entwicklungen bspw. durch die Einrichtung komfortabler Such- und Recherchemöglichkeiten wie etwa bei Wilhelm, Schorn (1999). Eine IT-getriebene Modifikation der Untersuchungssystematik findet sich bei Dobry (2003), indem ein Matrixanalyse-Modul vorgestellt wird, das eine Verknüpfung von Funktions-, Fehler- und Systemstruktur ermöglicht. Ein Ansatz zur automatischen FMEA-Erstellung findet sich in Hughes, Chou et al. (1999).

Beiträge der Forschung hinsichtlich einer *IT-basierten Modifikation des Untersuchungsobjekts* bei der FMEA-Erstellung und einer Wiederverwendung des Wissens einer FMEA für andere Anwendungen finden sich als Ansätze bspw. zur Entwicklung Wissensbasierter Systeme für die Störungsdiagnose bei Timpe, Seibicke (2000) und Struss, Heller et al. (2000) und als Ansätze zur Anbindung weiterer Informationsressourcen bspw. bei Huang, Shi et al. (2000).

3.1.4.3 IT-Unterstützung

Der Einsatz von IT bei der Durchführung einer FMEA wird als positiv angesehen.[240] Bspw. können die Teammitglieder online den Diskussionsstand verfolgen, so Missverständnisse frühzeitig ausräumen und das Ergebnis mit weniger Nachbearbeitungsaufwand erreichen. Durch Zugriff auf bereits durchgeführte FMEAs können Wiederholungsfehler vermieden werden.[241]

Wie bereits erwähnt wurde, kann die IT-basierte FMEA-Erstellung nach drei Kategorien unterschieden werden:[242]

1. Softwareanwendungen ausschließlich zum Editieren von Formblättern,
2. Softwareanwendungen als eigenständige, spezialisierte FMEA-Module,
3. Softwareanwendungen im Rahmen eines CAQ-Systems.

Die erste Kategorie sollte in „modernen“ Verwendungen von Instrumenten des Qualitätsmanagements keine Rolle spielen, weil sie nicht weit über die Rolle der Informationstechnik als „Schreibmaschine“ hinausgeht. Sie wird den Anforderungen an wirtschaftswissenschaftliche Abläufe in keiner Weise mehr gerecht, weil es allgemein als notwendig erachtet wird, dass „höher“ entwickelte Softwaresysteme, die über eine bloße Editierfunktion von Text hinausgehen, Verwendung finden, damit eine systematische Vorgehensweise und die Wiederverwendung von Wissen ermöglicht wird.[243] Deshalb wird die erste Kategorie nicht weiter untersucht. Die beiden letztgenannten Kategorien werden als FMEA-Anwendungen zusammenge-

240) Vgl. DGQ-Band 4 (2003), S. 34; Nickel (1992), S. 19 ff.; Paulic, Starke (1994), S. 34 ff., und Witter (1995), S. 39.

241) Vgl. DGQ-Band 13-11 (2001), S. 68.

242) Vgl. DGQ-Band 13-11 (2001), S. 68.

243) Vgl. Nickel (1992), S. 20 ff.

fasst. Dabei sind die folgenden Hard-/Softwarekonzepte und Funktionalitäten zurzeit als Standard anzusehen.[244]

Hardware-/Softwarekonzepte:

- Systeme funktionieren nach dem Client-Server-Prinzip,
- sind damit netzwerkfähig und
- unterstützen Datenbankanbindungen.

Funktionalitäten:

- Die Anforderungen nach VDA 4.2 und QS 9000 werden erfüllt,
- oft wird ermöglicht, zwischen System-, Konstruktions- und Prozess-FMEA bei der Durchführung zu unterschieden,
- verschiedene spezifische Formblatttypen werden unterstützt (VDA ´86 und ´96, herstellerspezifische Formblätter bspw. von Bosch und Ford),
- die Erstellung einer FMEA erfolgt über Fehlernetze oder die direkte Eingabe in das Formblatt,
- eine Katalogunterstützung zu den Formblattspalten wird geboten,
- es findet sich eine Auswertungs- und Überwachungsfunktion,
- VDA- und benutzerdefinierte Bewertungstabellen für die Risikoanalyse werden unterstützt,
- eine Versionenverwaltung und
- eine Benutzerverwaltung werden ermöglicht.

---

244) Vgl. Schloske (1999), S. 46. Eigene Recherchen des Verfassers führten zu der Feststellung, dass sich seit 1999 keine wesentlichen Veränderungen sowohl bei den Hard-/Softwarekonzepten als auch bei den Funktionalitäten der am Markt befindlichen Systeme vollzogen haben. Innerhalb der Recherchen, die kontinuierlich bis zum 8.05.2006 vorgenommen und vornehmlich über das Internet durchgeführt wurden (der Verfasser folgte hier der Prämisse, dass Unternehmen, die die Entwicklung und den Vertrieb von Software als Geschäftszweck haben, in der aktuellen Wettbewerbssituation nicht umhin können, über das Internet verfügbar zu sein, schon allein um ihre Technikkompetenz unter Beweis zu stellen), wurden die folgenden FMEA-Anwendungen berücksichtigt:
- SCIO-FMEA der Plato AG (http://www.plato-ag.com),
- CIMOS FMEA 7.0 der MBFG GmbH & Co. KG (http://www.irmler.com),
- FMEAworx der Eigenworx Company, Ltd. (http://www.eigenworx.com),
- Xfmea der ReliaSoft Corporation (http://www.reliasoft.com),
- Relex FMEA/FMECA der Relex Software Corporation (http://www.relexsoftware.com),
- FS [FMEA] der CAQ AG Factory Systems (http://www.caq.de),
- FMEA-Pro 7 der Dyadem International Ltd. (http://dyadem.com),
- APIS IQ-FMEA der Apis Informationstechnologien GmbH (http://www.apis.de),
(alle letzter Zugriff am 8.12.2006).

Am Markt befindliche Systeme werben mit dem Schlagwort „Wissensbasiert“.[245] Dieses Schlagwort bezieht sich jedoch lediglich auf die Wiederverwendung bereits erstellter und gespeicherter FMEA-Informationen. Implizites Wissen wird hingegen nicht explizit gemacht und auch nicht von der Anwendungssoftware (als Akteur) zur Problemlösung eingesetzt.[246] Es wird deutlich, dass die derzeit verfügbaren FMEA-Anwendungen nicht über eine datenbankgestützte Verarbeitung hinausgehen. Zwar gab es in der Forschung einige Versuche, mit Wissensbasierten Systemen eine FMEA-Erstellung zu ermöglichen[247]; es lässt sich jedoch nicht feststellen, ob diese überhaupt jemals einer „aktiven“ Verwendung zugeführt wurden. Hier eröffnet sich eine Umsetzungslücke, die noch zu schließen ist.

245) Beispielsweise wird mit diesem Schlagwort für die Anwendungssoftware SCIO-FMEA der Plato AG (http://www.plato-ag.com/platohp/scio-fmea.html); die Anwendungssoftware FS [FMEA] der CAQ AG Factory Systems (http://www.caq.de/de/products/product.asp?product=08); die Anwendungssoftware FMEAworx der Eigenworx Co., Ltd. (http://www.eigenworx.com/eigenworx_ko/ main/index.html) und die Anwendungssoftware CIMOS FMEA 7.0 der MBFG GmbH & Co. KG (http://www.irmler.com/ fmeasoft.pdf) geworben (alle Zugriff am 8.09.2006).

246) Siehe hierzu auch die Ausführungen zu *Wissen* und *Wissensrepräsentation* in den Kapiteln 2.1.2.1 und 2.1.2.2, S. 22 bzw. 23.

247) So finden sich Beispiele in Berthold, Krämer et al. (1995); FQS (1994); Nedeß, Nickel (1993); Wirth, Berthold et al. (1996).

3.2 Repräsentation von Wissen

3.2.1 Wissen einer FMEA

Unternehmenswichtiges Wissen[248] über fehlerrelevante Zusammenhänge, über das oft nur wenige Experten verfügen, wird in FMEA-Formularen gesammelt.[249] Oftmals stellt die FMEA sogar die erste Explikation solch implizit vorhandenen Wissens dar. Das vorhandene Expertenwissen zum Vorteil der FMEA[250] zielgerichtet zu nutzen, induziert die Entwicklung eines Wissensbasierten Systems zur weiteren Nutzung von Wissen.[251] Um dieses Wissensbasierte System zu entwickeln, muss zunächst die Repräsentation des Wissens einer FMEA untersucht werden. Hierbei ergeben sich zwei Untersuchungsfelder. Zum einen ist Wissen in der Durchführung einer FMEA enthalten (prozedurales Wissen) und zum anderen ist Wissen in den Ergebnissen nach einer Durchführung enthalten, d. h. Wissen ist in den Formblättern einer FMEA enthalten (deklaratives Wissen).[252]

Um im anschließenden Unterkapitel das Wissen einer FMEA weiter zu explizieren, werden zunächst der Ablauf zur Durchführung einer FMEA und die grundlegenden Informationen der Formblätter als Elemente des Wissens einer FMEA identifiziert und die Repräsentation von Wissen darin untersucht.

3.2.2 Wissen in der Durchführung einer FMEA

Abbildung 13 zeigt den strukturierten und arbeitsteiligen Ablauf zur Durchführung einer FMEA von Redeker, Sauer et al.[253] in Anlehnung an die von der VDA vorgegebene Richtlinie mit ihren fünf Schritten.[254] Insbesondere wird in der vorliegenden Prozessgestaltung auf die

248) Zum Begriff „Wissen“ siehe Kap. 2.1.2.1, S. 22 f.

249) Vgl. FQS (1994), S. 13. Die Begriffe „FMEA-Formblatt“ und „FMEA-Formular“ werden synonym verwendet.
Im Gegensatz zu den im vorherigen Kapitel genannten FMEA-Softwaresystemen, die als nicht wissensbasiert bezeichnet wurden, wird ein Wissensbasiertes System auf der Grundlage von FMEA-Formblättern in die Lage versetzt, implizites Wissen zu explizieren und als Akteur dieses Wissen zur Problemlösung einzusetzen (bspw. in dem mittels Regeln aus dem Verhalten von bereits hinterlegten Prozessabläufen auf das Verhalten zu untersuchender Prozessabläufe geschlossen werden kann). Siehe hierzu auch das Kapitel 8, S. 252 ff.

250) Vgl. DGQ-Band 13-11 (2001), S. 9.

251) In der Literatur findet sich vereinzelt der Hinweis, dass das Instrument FMEA geeignet sei, eine Wissensbasis aufzubauen (vgl. DGQ-Band 13-11 (2001), S. 9). Auch Struß verweist darauf, dass Wissen aus einer FMEA nutzbar gemacht werden kann, um Diagnose-Systeme zu entwickeln (vgl. Struss, Heller et al. (2000), S. 24).

252) Der Begriff „enthalten“ bezieht sich in diesem Fall auch auf die Möglichkeit, dass implizites Wissen abgeleitet und somit expliziert werden kann.

253) Vgl. Redeker, Sauer et al. (2002), S. 1030.

254) Vgl. VDA-Band 4 (2003), S. 15. Zur Begründung der Verwendung der VDA-Richtlinie siehe nächstes Kapitel.

Aufgabenverteilung eingegangen. Die Autoren schlagen einen Ablauf vor, der möglichst seltene zeitliche Zusammenkünfte von bereichsübergreifenden Teams berücksichtigt.[255]

Die Phase *Vorbereitung*, in der sich Planer und Moderator zur Vorbereitung der ersten Teamsitzung treffen, umfasst die Festlegung des Untersuchungsbereichs einschließlich der Zusammenstellung von Planungsunterlagen und der Planung von Zeitumfang, Teamgröße und Teamzusammensetzung. Noch in der Vorbereitung werden die Systemelemente und ihre Systemstruktur in einer Systemanalyse sowie eine erste grobe Funktionsstruktur als Orientierungshilfe festgelegt.

Es folgt *Teamphase I*. Während der Teamphase I soll zwischen den Teammitgliedern in der Funktionsanalyse Konsens erlangt werden hinsichtlich einer verbindlichen Funktionsstruktur, die auf der Systemstruktur basiert. Hierzu wird die anfänglich als Orientierungshilfe dienende Funktionsstruktur ergänzt und abschließend festgelegt. Im Anschluß wird zur Fehleranalyse übergegangen und die Fehlfunktionsstruktur ermittelt.[256] Eine Fehleranalyse mit der Festlegung von Fehlerfolgen, Fehlern und Fehlerursachen wird durchgeführt. Am Ende der Teamphase I liegt in der Regel gegenüber der aus der Phase Vorbereitung stammenden ursprünglichen Funktionsstruktur eine komplett veränderte Funktionsstruktur vor. In Einzelfällen kann es sogar notwendig werden, die Systemstruktur zu verändern.

Die ermittelten System-, Funktions- und Fehlfunktionsstrukturen werden in der Phase *Aufbereitung* IT-basiert nachbearbeitet und in Formblätter eingegeben.

Anschließend beginnt die *Teamphase II*. Hier erfolgt die eigentliche Formblattarbeit im Team. Eine Risikobewertung der möglichen Fehler wird durchgeführt. Es werden Maßnahmen zur Vermeidung und Entdeckung ermittelt. Termine und Verantwortlichkeiten für die Verwirklichung der Maßnahmen werden festgelegt. Nach der Optimierung erfolgt eine erneute Risikobewertung.

In der Phase der *Nachbearbeitung* wird seitens des Moderators die FMEA-Durchführung mit einer Auswertung der erzielten Ergebnisse und Erstellung einer umfassenden Dokumentation abgeschlossen.

255) Vgl. Redeker, Sauer et al. (2002), S. 1030.

256) Es kommt zu einer bewussten Überlappung in der Teamphase I von Funktions- und Fehleranalyse, d. h. die Mitglieder des FMEA-Teams bearbeiten Funktions- und Fehleranalyse gemeinsam, bevor der Moderator anschließend (allein) eine Aufbereitung der Fehleranalyse vornimmt.

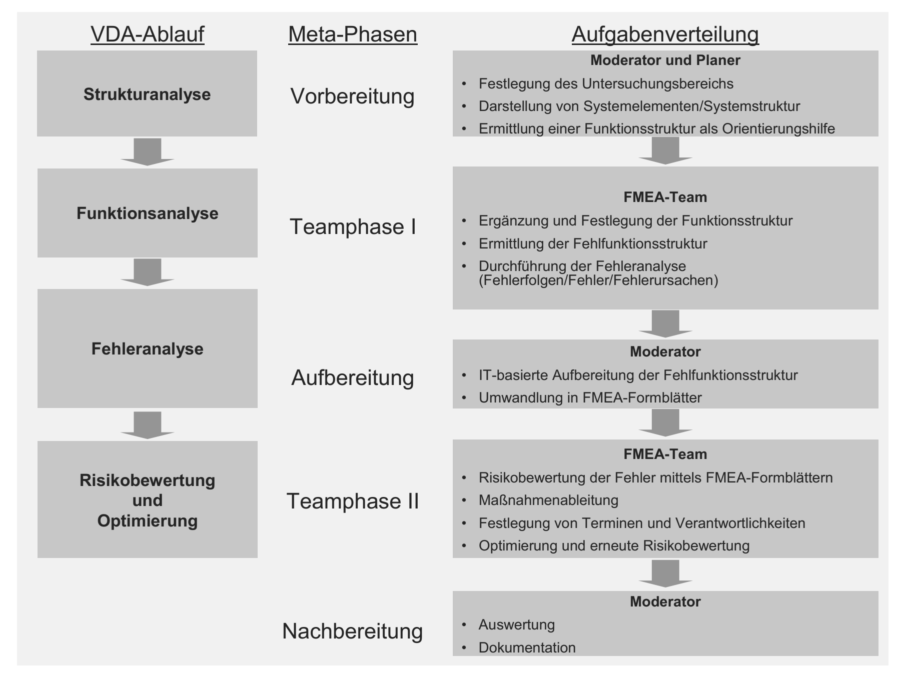


Abbildung 13: Herkömmliche Aufgabenverteilung bei FMEA-Erstellung[257]

3.2.3 Wissen in den einzelnen Hauptelementen einer FMEA

Ein FMEA-Formblatt lässt sich in fünf Hauptelemente gliedern, die aufeinander aufbauend vorliegen (sollten). Diese Hauptelemente einer FMEA sind:[258]

1. Kopfzeile (auch Header),
2. Systemstruktur (auch systembeschreibende Struktur),
3. Funktionsstruktur,
4. Fehlerstruktur (auch Fehlfunktionsstruktur) und
5. Maßnahmenstruktur.

Die Abbildung 14 verdeutlicht diesen „konzeptionellen“ Blick auf ein FMEA-Formblatt. Die *Kopfzeile* enthält globale Angaben zur Untersuchung und dem untersuchten System. Die *Sys-*

257) In Anlehnung an Redeker, Sauer et al. (2002), S. 1030.

258) Abweichend hierzu verzichtet Eberhardt (2003), S. 83 ff., auf die Funktionsstruktur als Hauptelement. Vom Verfasser wird diese jedoch, wie auch in den meisten anderen Quellen (bspw. DGQ-Band 13-11 (2001), S. 28; FQS (1994), S. 21; Schloske (1999), S. 31 ff.), sehr wohl als Hauptelement aufgefasst, weil die Fehlerstruktur auf ihr basiert.

temstruktur bildet das zu untersuchende System ab. Sie lässt sich dabei vielfältig unterscheiden hinsichtlich möglicher Betrachtungseinheiten (bspw. in produkt- oder prozessorientierter Sicht). Die *Funktionsstruktur* besteht im Wesentlichen aus der Ermittlung und Zuordnung von Funktionen zu den Elementen der Systemstruktur. Für jedes Systemelement und seine dazugehörigen Funktionen wird ein Formblatt angefertigt. Die *Fehlerstruktur* lässt sich als Kombination der Unterelemente Fehlerursache, Fehler und Fehlerfolge im Sinne einer Ursache-Wirkungs-Kette darstellen, die auf die berücksichtigten Funktionen zurückgreift, indem sie mit diesen Funktionen verknüpft wird. Eine *Maßnahmenstruktur*, die der Ermittlung und Vorhaltung von Optimierungsmaßnahmen dient, lässt sich fehlerbezogen unterscheiden nach Vermeidungs- und Entdeckungsmaßnahmen. Die Maßnahmenstruktur wird wesentlich beeinflusst durch die Bewertungsergebnisse bei der Ermittlung der Risikoprioritätszahlen.

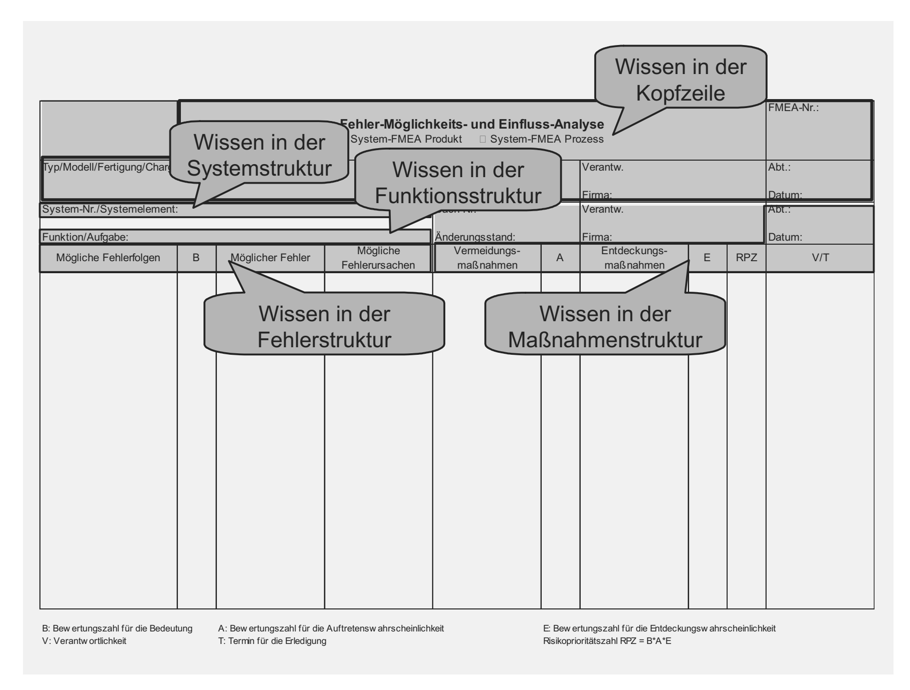


Abbildung 14: Wissen in den Hauptelementen eines FMEA-Formblatts

Über die Automobilindustrie hinaus hat sich die Vorgabe des VDA 4.1 (System-FMEA-Formblatt VDA ´96) als Standard etabliert, so dass im weiteren Verlauf der Untersuchungen als gemeinsamer Nenner dieser Standard Verwendung findet.[259]

259) Eine ähnliche Argumentation findet sich bspw. auch in Schöfer (1999), S. 106. Prinzipiell ist es für die Erkenntnisse dieser Arbeit unerheblich, welcher Standard für die Entwicklung einer FMEA-Ontologie, die wiederum im Wissensbasierten System OntoFMEA eingesetzt wird, Verwendung findet. Das OntoFMEA-Vorgehensmodell dieser Arbeit unterstützt in seiner Generizität die unterschiedlichsten Standards. Nichtsdestotrotz erscheint es sinnvoll, sich auf einem gemeinsamen weit verbreiteten Standard festzulegen, um die Nachvollziehbarkeit der Argumentation leichter zu ermöglichen.

Grundlage dieser Entscheidung war:

- im deutschsprachigen Raum gilt der VDA-Standard als am weitesten verbreitet, weil er maßgeblich in der deutschen Automobilindustrie Verwendung findet;[260]
- die weitaus meisten computergestützten FMEA-Anwendungen unterstützen diesen Standard;[261]
- sämtliche verfügbaren Standards ähneln weitestgehend einander, so dass die Fokussierung auf einen Standard hauptsächlich der Nachvollziehbarkeit dieser Arbeit durch den Rezipienten mittels Festlegung auf bestimmte Begrifflichkeiten dient;
- in seiner aktuellsten Ausgabe[262] stellt er neben dem SAE-Standard[263] den „modernsten" Ansatz dar.[264]

260) Vgl. DGQ-Band 13-11 (2001), S. 24; Pfeifer (2002), S. 382 f.; Redeker, Sauer et al. (2002), S. 1030; Witter (1995), S. 15.

261) So bspw. die Systeme APIS IQ-FMEA, CIMOS FMEA 7.0, FS [FMEA] und SCIO-FMEA. Siehe hierzu auch die Fn. 244, S. 55.

262) Vgl. VDA-Band 4 (2003).

263) Vgl. SAE J1379:2002.

264) Dabei geht der Verfasser von der verkürzenden, in diesem Zusammenhang jedoch als Arbeitsthese plausiblen Annahme aus, dass sich in diesem Faktum auch die elaborierteste Anwendung einer FMEA wiederfindet.

3.3 Explikation von Wissen in einer FMEA

Die genannten Hauptelemente werden eingehender betrachtet im Sinne eines Explikationsvorgangs, der zum Ziel hat, einige wichtige Punkte, die implizit bei der Durchführung einer FMEA enthalten sind, aufzuzeigen und diese für die späteren Untersuchungen vorzubereiten. Die hier vorgestellten „Sachverhalte" werden später bei der Entwicklung der FMEA-Ontologie und des Vorgehensmodells gebraucht.[265]

3.3.1 Wissen in der Kopfzeile

Die Kopfzeile eines FMEA-Formblatts enthält die Stammdaten zu einem Untersuchungsgegenstand. Aus den Stammdaten soll der betrachtete Untersuchungsgegenstand zu erkennen sein.[266] Diese Stammdaten umfassen gemäß VDA-Formblatt:

- FMEA-Art,
- FMEA-Nummer,
- Typ/Modell/Fertigung/Charge,
- Sach-Nummer/Änderungsstand,
- Verantwortlicher/Firma[267] und
- Abteilung/Datum.

Bei jeder FMEA wird festgelegt, welcher Art sie entspricht, d. h., ob es sich um eine System-FMEA Produkt oder System-FMEA-Prozess handelt. Ferner bekommt jede FMEA eine identifizierende Nummer zugewiesen. Mittels präziser Stichwortangaben wird kurz benannt, um welchen Typ oder welches Modell, welche Fertigung oder welche Charge es sich beim Untersuchungsgegenstand handelt. Existiert für das untersuchte System eine eindeutige Nummer, die deckungsgleich mit der Identifikationsnummer einer Stückliste zum Untersuchungsgegenstand ist, dann wird diese als Sach-Nummer mit dem jeweiligen Änderungsstand angegeben. Es werden ein verantwortlicher Mitarbeiter, dessen Unternehmen und seine Abteilung namentlich genannt. Abschließend wird der FMEA ein Datum des Beginns der Durchführung zugeordnet.

---

265) Insbesondere bei der Entwicklung der Hauptkonzepte der Meta-Ebene (Kapitel 7.3.3, S. 232 ff.) und bei der Entwicklung der Taxonomien für Funktion, Systemelement und Maßnahme (Kapitel 7.4, S. 244) sowie bei der Entwicklung des Vorgehensmodells (Kapitel 6, S. 193 ff.) werden die hier explizierten Sachverhalte einer FMEA einer Verwendung zugeführt.

266) Vgl. VDA-Band 4 (2003), S. 27.

267) An dieser Stelle zeigt sich die Perspektive des „Ingenieurs" beim VDA-Formblatt im Original. In den folgenden Ausführungen wird der betriebswirtschaftlich *passendere* Begriff „Unternehmen" verwendet, weil bei den Stammdaten das zugehörige Unternehmen als Verantwortungsträger und nicht dessen juristische Bezeichnung genannt werden soll.

3.3.2 Wissen in der Systemstruktur

Um die strukturellen Zusammenhänge in einem System darzustellen, wird ein System in Systemelemente zerlegt, die hierarchisch angeordnet werden. Die systembeschreibenden Strukturen bestehen somit aus einem hierarchischen Aufbau von Systemelementen. Grundsätzlich lässt sich dabei die systembeschreibende Struktur anhand der hier beispielhaft untersuchten Vorgabe einer System-FMEA in eine produktorientierte Struktur und eine prozessorientierte Struktur unterteilen.[268] Die einzelnen Formblätter, die jeweils für die einzelnen Systemelemente und deren zugeordneten Funktionen angefertigt werden, beziehen sich jeweils auf ein(en) Typ/Modell/Fertigung/Charge und ergeben in ihrer Gesamtheit die Systemstruktur.

Spätestens jedoch bei der weiteren Untergliederung des zu untersuchenden Systems wird deutlich, dass die Systemstruktur vor allem von der Art der Betrachtung abhängt und auf vielfältige Weise dargestellt werden kann. Bspw. kann ein Motor hinsichtlich seiner Funktionseinheiten (Benzin-, Abgas- und Öllauf) oder seiner Baugruppen (Kolben, Einspritzanlage) betrachtet werden. Zusätzlich ist es möglich, die „Aggregationsstufe" des zu untersuchenden Systems zu variieren. Bspw. kann der Motor als Solitär untersucht werden oder als Teil eines Ganzen (Pkw). In jedem Fall wird der Untersuchungsgegenstand als *Betrachtungseinheit* aufgefasst.[269] Eine Betrachtungseinheit entspricht dabei einem nach Aufgabe und Umfang abgegrenzten Gegenstand einer Betrachtung.[270]

Eine Betrachtungseinheit wird hinsichtlich der zwei Dimensionen *Betrachtungsebene* und *Betrachtungskriterium* unterschieden.

Die Betrachtungsebenen werden innerhalb eines hierarchischen Aufbaus in vier Ebenen unterteilt:[271]

1. System
   Als System wird die Gesamtheit der erforderlichen Mittel zur eigenständigen Erfüllung eines Aufgabenkomplexes angesehen. Diese Ebene ist die oberste Betrachtungsebene.

---

268) Im folgenden Text wird nur auf die produktorientierte Struktur näher eingegangen, weil sie dem Fokus der Untersuchung dieser Arbeit entspricht. Die prozessorientierte Struktur wird allgemein verwendet bei der Untersuchung von Produktionsabläufen, Bearbeitungsabläufen und Bedienungsabläufen (vgl. Eberhardt (2003), S. 93). Die Fehleranalyse findet ihren Fokus in diesem Fall in den Tätigkeiten, die den betrieblichen Ablauf prägen.

269) Alternativ ließe sich ein Untersuchungsgegenstand bspw. als Ort im Raum auffassen, d. h. der Gegenstand wird räumlich anhand eines festgelegten Koordinatensystems untersucht. Die hier verwendete Auffassung als Betrachtungseinheit fügt sich in die Ausführungen zur Anwendung von Ontologien (siehe bspw. Kapitel 4.1.4, S. 86 ff.) ein, bei denen die Betrachtungsleistungen des „Ontologie-"Entwicklers betont werden, und wird deshalb an dieser Stelle eingeführt.

270) Vgl. DIN 40150:1979, Kapitel 2, Seite 1. Die DIN 25448:1990 nennt alternativ für den Begriff „Betrachtungseinheit" auch den Begriff „Baueinheit". In der Ausgabe von 1980 wurde hier noch der Begriff „Komponente" verwendet.

271) Vgl. DIN 40150:1979, Kapitel 2.2, S. 1.

2. Einrichtung
   Eine Einrichtung ist eine eigenständig verwendbare Betrachtungseinheit.[272] Sie stellt eine Zusammenfassung der beiden folgenden Ebenen von Elementen und/oder Gruppen dar.

3. Gruppe
   Eine Gruppe ist eine Zusammenfassung von Elementen der vierten Ebene zu noch nicht eigenständig verwendbaren Betrachtungseinheiten.

4. Element
   Ein Element wird als unteilbare Einheit der untersten Betrachtungsebene angesehen.

Zwischen der obersten und der untersten Ebene lassen sich beliebig viele weitere Ebenen definieren. Hier kann keine starre Vorgabe, die für alle zu untersuchenden Systeme einen allgemeingültigen Charakter aufweist, gegeben werden.

Ein Betrachtungskriterium legt die Merkmale fest, anhand derer das zu untersuchende System in einer Systemstruktur modelliert werden soll. Zumeist lassen sich Betrachtungseinheiten anhand der grundsätzlichen Kriterien *Funktionseinheit* oder *Baueinheit* unterscheiden. Eine Funktionseinheit entspricht dabei einer Betrachtungseinheit, die nach ihrer Aufgabe abgegrenzt wird. Eine Baueinheit wird hingegen nach ihrem Aufbau abgegrenzt.[273]

Mit Hilfe der beiden Dimensionen lassen sich Betrachtungseinheiten in einer Matrix den verschiedenen Betrachtungsebenen nach bestimmten Betrachtungskriterien zuordnen. Die nachstehende Tabelle 2 verdeutlicht die mögliche Matrix.[274]

---

272) Bspw. lässt sich bei einem handelsüblichen Heimcomputer (als System) eine Grafikkarte als eine eigenständig verwendbare Betrachtungseinheit auffassen, denn diese Grafikkarte kann in einem weiteren Heimcomputer verwendet werden. Betrachtet man jedoch das System „Heimcomputer" ohne diese Grafikkarte, so erscheint es einleuchtend, dass der Aufgabenkomplex desselben nicht erfüllt werden kann, weil nicht die Möglichkeit besteht, durchgeführte Rechenoperationen zu visualisieren.

273) Alternativ können andere Betrachtungskriterien verwendet werden. Bspw. kann eine Betrachtungseinheit anhand der Kriterien *Betriebseinheit*, *Instandhaltungseinheit*, *Prüfeinheit*, *Austauscheinheit* und *Fertigungseinheit*, die als „Meta-Funktionen" aufgefasst werden können, unterschieden werden.

274) Beispiele hierzu finden sich in der DIN 40150:1979, Kap. 4, S. 3.

Betrachtungs-kriterien / Betrachtungsebenen	Funktionseinheit	Baueinheit	Betriebseinheit	Instandhaltungs-einheit
Element	Funktionselement	Bauelement	Betriebselement	Instandhaltungs element
Gruppe	Funktionsgruppe	Baugruppe	…	…
Einrichtung	funktionale Einrichtung	bauliche Einrichtung (Gerät, Anlage)	…	…
System	funktional abgegrenztes System	baulich abgegrenztes System	…	…

Tabelle 2: Betrachtungskriterien und -ebenen gemäß DIN 40150:1979

Wann eine Betrachtungseinheit als elementar anzusehen ist, liegt im Ermessensspielraum der ausführenden Analysten (FMEA-Teammitglieder). Hierbei ist zwischen den Komponenten, zwischen denen ein Trade-off hinsichtlich Aufwand (bspw. in Personenstunden) und Vollständigkeit (bspw. denkmöglicher Funktionen) besteht, zu wählen.

Bei der Durchführung einer FMEA bietet es sich zumeist an, das Betrachtungskriterium „Funktionseinheit" einzusetzen, weil im nächsten Durchführungsschritt (Teamphase I) die Funktionsanalyse ansteht und so eine bestmögliche Anschlussfähigkeit erreicht wird. Jedoch lässt sich auch mit dem Kriterium „Baueinheit" und der anschließenden Ermittlung von Funktionen das gewünschte Ziel einer vollständigen Funktionsanalyse erreichen.

3.3.3 Wissen in der Funktionsstruktur

Für das betrachtete System werden Funktionen definiert und den Systemelementen zugeordnet. Es empfiehlt sich, eine Funktion mittels eines Objekts und eines Verbs zu beschreiben (z. B. Schraube fixieren).[275] Prinzipiell lassen sich hierbei zur Arbeitserleichterung Rückgriffe auf existierende Kataloge tätigen, die bspw. technische Funktionen alphabetisch sortiert vorhalten.[276]

275) Vgl. DGQ-Band 13-11 (2001), S. 28

276) Siehe hierzu Kapitel 7.4.3, S. 247 ff.

Zur Vorbereitung auf die Fehleranalyse werden die ermittelten Funktionen in ihren funktionalen Abhängigkeiten abgebildet. Dabei werden Funktionen zu übergeordneten Funktionen zusammengefasst oder als Teilfunktionen untergeordnet.

Als Synthese aus dem Vorgehen zur Entwicklung einer System- und einer Funktionsstruktur schlägt Nickel eine funktionsorientierte Strukturanalyse[277] in fünf Schritten vor:[278]

1. Systemdefinition
   Mit der Systemdefinition wird der Umfang der Betrachtungseinheit festgelegt.

2. Externe Funktionen
   Die zu erfüllenden Aufgaben gemäß der Systemgrenzen werden bestimmt.[279]

3. Interne Funktionen
   Es werden Aufgaben für die internen Funktionen ermittelt. Die internen Funktionen ermöglichen die externen Funktionen.[280]

4. Funktionselemente
   Es werden die Funktionselemente bestimmt, die die internen Funktionen erfüllen.[281]

5. Merkmale
   Funktionskritische Merkmale der Funktionselemente werden bestimmt.[282]

Der letzte Schritt dient zur Vorbereitung auf die anstehende Fehleranalyse. Besonders die funktionskritischen Merkmale von Funktionselementen müssen bei der Durchführung einer FMEA einer Fehleranalyse und einer Risikobewertung unterzogen werden, weil die Nichterfüllung einer Anforderung, die einen Funktionsausfall zur Folge hat, mittels des präventiven Instruments FMEA verhindert werden soll. Durch die Konzentration eines FMEA-Teams auf

277) Hingegen schlägt der VDA ein sukzessives Vorgehen für Struktur- und Funktionsanalyse vor, d. h., erst nach einer vollständig angelegten Systemstruktur erfolgt die Funktionsstrukturermittlung (vgl. VDA-Band 4 (2003), S. 16 f.).

278) Vgl. Nickel (1992), S. 41.

279) Die zu untersuchende Betrachtungseinheit wird als Black-Box aufgefasst, deren innere Abläufe dem Betrachter verborgen bleiben. Lediglich die Aufgaben, die von der Betrachtungseinheit übernommen werden und den Zweck der Betrachtungseinheit für einen Benutzer ausmachen, werden als externe Funktionen wahrgenommen. Bspw. wird die Aufgabe *Antrieb eines Fahrzeugs* (das Fahrzeug sei die Black-Box) als Bewegung des Fahrzeugs und damit als externe Funktion wahrgenommen.

280) Spinnt man den Gedankengang der vorherigen Fußnote weiter, so lässt sich für die Black-Box und deren externe Funktionen feststellen, dass selbige Funktionen durch interne Funktionen bereitgestellt werden müssen. Diese internen Funktionen müssen Aufgaben erfüllen, die die Aufgaben der externen Funktionen ermöglichen. Bspw. wird die Funktion der Kupplung als die Aufgabe, den Kraftschluss zwischen Motor und Getriebe herzustellen, innerhalb der Black-Box als interne Funktion festgelegt. Diese interne Funktion wird gebraucht, um die externe Funktion Antrieb eines Fahrzeugs zu erfüllen. Es wird deutlich, dass mehrere interne Funktionen zur Erfüllung einer externen Funktion benötigt werden können. Bspw. wird noch ein Drehmoment für den Antrieb benötigt.

281) Bspw. enthält die interne Funktion *Kraftschluss herstellen* als mögliche Elemente ihrer Funktion *einkuppeln* und *auskuppeln*.

282) Als „funktionskritisch“ sind Merkmale einzustufen, die bei ihrer Nichterfüllung den Ausfall der Funktion zur Folge haben.

die funktionskritischen Merkmale lässt sich ein Vorgehen „mit dem Blick für das Wesentliche“ einhalten.

3.3.4 Wissen in der Fehlerstruktur

Aus den Angaben in Kapitel 3.1.4.1.3, S. 49 ff., und den vorangegangen zwei Unterkapiteln folgt, dass je nach Betrachtungsebene ein Fehler auf einer höheren Betrachtungsebene zu einer Fehlerursache und auf einer niedrigeren Betrachtungsebene zu einer Fehlerfolge werden kann, weil jede Betrachtungseinheit auf unterschiedlichen Betrachtungsebenen untersucht werden kann. Die unterschiedlichen Betrachtungsebenen haben bspw. zur Folge, dass ein Fehler, der einer internen Funktion zugeordnet wurde, als Fehlerursache einer externen Funktion angesehen wird.[283] Die Abbildung 15 auf der folgenden Seite verdeutlicht diesen Zusammenhang. Ein Gesamtsystem (Schraubmaschine) kann für den Fehler „Maschinenstörung“ als Fehlerursache ein defektes Bedienpult haben. Das Bedienpult lässt sich auf der Baugruppenebene (Ebene 2) zu anderen Systemelementen (wie etwa dem Transportband) abgrenzen. Das defekte Bedienpult als Fehler hat auf dieser Ebene die mögliche Fehlerursache, dass der Stromkreislauf nicht geschlossen ist. Der Fehler „Bedienpult defekt“ kann auf einer noch tiefer gelegenen Ebene (Element „Stromkabel“) als Fehlerfolge des Fehlers „Stromkreislauf nicht geschlossen“ aufgefasst werden.

Während auf der Ebene 1 die Systemelementgrenze (SE-Grenze)[284] durch das System vorgegeben wird, wird auf der Ebene 2 die Systemelementgrenze anhand der einzelnen Baugruppen und ihrer Schnittstellen festgelegt. Auf der dritten Ebene werden die Systemelementgrenzen durch die einzelnen Bauelemente vorgeschrieben. Die Berücksichtigung dieser Überlappungen zwischen den Ebenen einer System-FMEA ist eine Grundvoraussetzung für eine durchgängige Analyse komplexer Systeme.[285]

---

283) Siehe hierzu insbesondere die Black-Box-Erläuterung in den Fn. 279 und 280, S. 66.

284) Die Systemelementgrenze wird ausgehend von der untersten Ebene der Betrachtungen (Elementebene) für die weiteren Ebenen festgelegt. Es wird dabei implizit davon ausgegangen, dass bspw. ein Stromkabel, das zu einem Bedienpult gehört, welches wiederum zu einer Schraubmaschine gehört, immer auch die Grenzen der höheren Aggregationsstufen definiert. Wird seitens des Betrachters beim Ebenenwechsel nicht auch ein Perspektivwechsel eingenommen, dann determinieren die Elementgrenzen mit ihrem spezifischen Aufbau immer auch die Grenzen der „gröberen“ Ebenen (bspw. des Gesamtsystems). Führt dieses Stromkabel bspw. an eine externe Stromquelle, so wird dieser Anschluß auch bei Betrachtung des (Gesamt-)Systems als Grenze des Systems wahrgenommen. Demgegenüber stehen mögliche interne Elementgrenzen, die für die Betrachtung auf der Systemebene nicht relevant sind, weil sie nicht bei einer Betrachtung auf Systemebene in Erscheinung treten (bspw. könnte das Stromkabel auch an eine systeminterne Quelle angeschlossen werden). Um diesen Sachverhalt ein wenig hervorzuheben und diese (internen) Elementgrenzen auszuschließen, wird im Folgenden der Begriff „Systemelementgrenze“ (SE-Grenze) verwendet.

285) VDA-Band 4 (2003), S. 25.

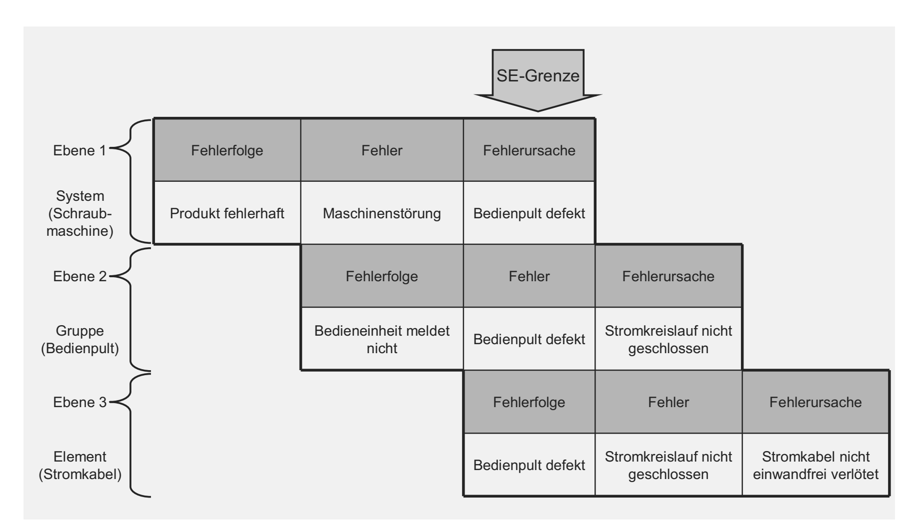


Abbildung 15: Fehlerstruktur und Ebenen der Betrachtungseinheit

Die einzelnen Möglichkeiten für Fehlerfolge, Fehler und Fehlerursache leiten sich zumeist aus den Funktionsstrukturen ab. Oftmals werden die enthaltenen Funktionen hinsichtlich ihrer Erfüllung negiert und so als Nicht-Erfüllung einer Anforderung gekennzeichnet.[286]

3.3.5 Wissen in der Maßnahmenstruktur

Prinzipiell bieten sich vier verschiedene Ansätze für Maßnahmen zur Fehlerbehandlung in der Phase der Optimierung:[287]

- Gänzliche Vermeidung der Fehlerursache,
- Reduzierung der Wahrscheinlichkeit des Auftretens eines Fehlers,
- Reduzierung der Bedeutung des Fehlers und
- Erhöhung der Wahrscheinlichkeit der Fehlerentdeckung.

Die ersten beiden Punkte stehen dabei in einem direkten Zusammenhang. Wenn Fehlerursachen vermieden werden, dann reduziert sich die Wahrscheinlichkeit des Auftretens auf den Erwartungswert Null. Wird bspw. ein bestimmtes Bauelement für einen Fehler als Ursache bestimmt, so lässt sich entweder diese Fehlerursache vermeiden, indem das Bauelement durch ein anderes Bauelement ersetzt wird, das dieselbe Funktionalität besitzt, jedoch von anderer Art ist. Oder es wird ein Bauelement von anderer Güte, jedoch von gleicher Art eingebaut,

286) Bspw. wird die Funktion „Stromkreislauf schließen" mittels Negation zum Fehler „Stromkreislauf nicht geschlossen".

287) Vgl. Kersten (1994), S. 477 f.

bspw. indem ein Bauelement eines anderen Herstellers verwendet wird, von dem bekannt ist, dass es eine längere Lebensdauer aufweist.[288]

Die Maßnahmen werden gemäß den meisten FMEA-Standards in Vermeidungs- und Entdeckungsmaßnahmen gegliedert.[289] Dabei lassen sich die ersten drei Punkte der Aufzählung (Vermeidung der Fehlerursache, Reduzierung der Auftretenswahrscheinlichkeit und der Bedeutung) als Vermeidungsmaßnahmen zusammenfassen.

288) In diesem Beispiel wird unterstellt, dass der Fehler in der zu geringen Lebensdauer eines Bauteils liegt. Andere Fehler sind selbstverständlich denkbar.

289) Vgl. bspw. DGQ-Band 13-11 (2001), S. 21; VDA-Band 4 (2003), S. 21.

3.4 Beispielhafte Anwendungsfälle

3.4.1 Karl Schumacher Maschinenbau GmbH

Für die weiteren Ausarbeitungen dieser Arbeit wird auf bereits ausgefüllte FMEA-Formblätter der Karl Schumacher Maschinenbau GmbH (KSM) aus Köln zurückgegriffen.[290] Die Formblätter entsprechen dabei den Vorgaben des VDA aus dem Jahr 1996. Es werden in den beispielhaften Ausführungen der folgenden Kapitel zur FMEA-Ontologie und zum Wissensbasierten System OntoFMEA Teile der Formblätter zweier Schraubmaschinen verwendet. Nach einer kurzen Darstellung der KSM wird deshalb kurz auf zwei Maschinen, für die die FMEAs durchgeführt wurden, näher eingegangen. Die hier verwendeten Formblätter finden sich im Anhang dieser Arbeit.[291]

Die KSM wurde 1964 gegründet. Das unabhängige, mittelständisch geprägte Unternehmen expandierte während seiner Entwicklung kontinuierlich. Zurzeit werden ca. zwei Dutzend Mitarbeiter bei einer Betriebsgröße von ca. 1.300 m^2 beschäftigt. Der ursprünglich regionale Kundenkreis konnte in den letzten Jahren international ausgeweitet werden, dabei wird vor allem für die Automobilindustrie produziert.

Die KSM ist ein typischer Auftragsfertiger. Sie entwickelt, konstruiert, fertigt und montiert vor allem Sondermaschinen unter Einsatz von technologisch anspruchsvollen Maschinenbau- und Steuerungskomponenten mit hohen Sicherheits- und Leistungsstandards. Der Einsatz moderner, computergestützter Planungs-, Organisations- und Controllingsysteme sowie eine zeitgemäße Fertigung auf CNC-Werkzeugmaschinen versetzen das Unternehmen in Verbindung mit seinem langjährig erworbenen Engineering-Know-how in die Lage, Problemlösungen auch zu außergewöhnlichen und sehr komplexen Aufgabenstellungen zu liefern. Durch qualitativ und funktionell hochwertige Ausführung der Maschinen und Anlagen genießt das Unternehmen in seinem Kundenkreis einen hervorragenden Ruf.

Die beiden vorgestellten Maschinen sind Teile einer Fertigungsstraße, auf der Getriebe für Automobile montiert werden. Die erste Maschine (5-Automatik-Spindelschrauber), genauer: Teile der zugehörigen FMEAs, dienen als „Wissensbasis“ für das vorgestellte System OntoFMEA und sind bei der Funktionsdarstellung von OntoFMEA (Kapitel 8, S. 252 ff.) bereits im System als Wissensbasis implementiert.[292] Die zweite Maschine wird als Anwendungsbeispiel genutzt, um Aspekte der Kodierung zu erläutern und Funktionalitäten von OntoFMEA darzustellen.

290) Für weitere Informationen siehe http://www.ksm-maschinenbau.de, Zugriff am 18.03.2007.

291) Siehe Anhang A.1, S. 332 ff. Im weiteren Verlauf der Arbeit wird das Wissen innerhalb der Formblätter so weiter verwendet, wie es vorgefunden wurde. Es wird dabei davon ausgegangen, dass die durchführenden Mitarbeiter sich auf die Begriffe der Domäne verständigt haben. Im Sinne eines Wissensingenieurs verarbeitet der Verfasser lediglich dieses vorgefundene Wissen.

292) Siehe hierzu auch die FMEA-Ontologie und ihre Wissensbasis im Anhang A.2, S. 336 ff.

3.4.2 5-Automatik-Spindelschrauber

Der 5-Automatik-Spindelschrauber dient zur Verschraubung eines Getriebes. Das Getriebe (als Werkstück) ist beim Durchfahren der Maschine auf einen Werkstückträger montiert. Im Aufbau besteht die Maschine aus zwei Stationen, einer Schraubstation und einer Schwenkstation für den Werkstückträger. Aus konstruktiven Gründen wird es an einer Stelle der Fertigung notwendig, das Werkstück zu schwenken, um mit den Bauteilmaßen die Maschinenstraße durchfahren zu können.[293] Die Schraubeinheit verschraubt gleichzeitig fünf Schrauben waagerecht in Bohrungen eines Getriebes in 5 Schritten:

1. Einfahren
   Das auf einem Werkstückträger montierte Getriebe wird auf dem Transportband in die Maschine eingefahren und gestoppt. Nachfolgende Werkstückträger werden am Vorstopper angehalten, damit sich zu jedem Zeitpunkt höchstens ein Werkstückträger mit Getriebe in der Maschine befindet.

2. Fixieren
   Nach dem Stoppen wird der Werkstückträger mittels einer Fixiereinheit arretiert, so dass sich das Werkstück (Getriebe) nicht mehr frei bewegen lässt und mit der Verschraubung begonnen werden kann.

3. Verschrauben
   Es müssen für die 5-Automatik-Schraubeinheit Schrauben „automatisch" zugeführt werden.[294] Hierzu werden Schrauben in einen Schwingsortierer gegeben. Dort werden die Schrauben der Form (Länge und Durchmesser) nach getrennt, in Reihen aufgestellt und in einen Linearförderer gegeben. Der Linearförderer gibt die Schrauben, die durch eine Schraubenvereinzelung getrennt werden, zu einem Schiebekamm. Hier wird mittels eines Servomotors der Schiebekamm vor und zurück bewegt, bis fünf Aussparungen mit Schrauben befüllt worden sind. Bei voller Beladung werden die Aussparungen nach unten geöffnet und die Schrauben werden über Schläuche auf eine Schraubenaufnahmeplatte, die die Schrauben entsprechend den Bohrungen am Werkstück anordnet, geleitet. Die Schraubenaufnahmeplatte wird unter die 5-Automatik-Spindelschraubeinheit geschoben und von den Schraubern mit Hilfe von Magneten aufgenommen. Nachdem die Schraubenaufnahmeplatte zurückgefahren wurde, können die Schrauben im Getriebe verschraubt werden. Die Schrauben werden nach festgelegten Vorgaben des Gebers über Drehwinkel und Drehmoment verschraubt.

4. Lösen
   Nach der erfolgreichen Verschraubung und der Rückkehr der Schraubeinheiten in ihre

---

293) In dieser Arbeit wird jedoch lediglich die Schraubstation (als Schraubeinheit) weiter betrachtet, damit ein hohes Maß an Wiederverwendbarkeit bei den Beispielen mit dem 9+1-Spindelschrauber erreicht wird.

294) Im Grunde erfolgt die Zuführung *semi*-automatisch, weil zuerst Mitarbeiter regelmäßig Schrauben auf ein Zuführband (Bunkerband) schütten müssen, das einen Bunker befüllt. Anschließend wird über eine Klappe eine bestimmte Menge Schrauben in einen Schwingsortierer abgegeben.

jeweiligen Ausgangspositionen werden Fixiereinheit und Maschinenstopper gelöst. Der Werkstückträger enthält einen Informationsträger, auf dem Informationen zum Fertigungsprozess abgespeichert werden. Dieser wird nach Durchführung der Verschraubung beschrieben.

5. Ausfahren
   Der Werkstückträger wird mit dem Getriebe aus der Maschine gefahren.



Abbildung 16: 5-Automatik-Spindelschrauber (Karl Schumacher Maschinenbau GmbH)

Über ein Bedienpult lässt sich ein Wechsel zwischen Automatik- und Handbetrieb der Maschine vornehmen. Im Handbetrieb können die beweglichen Bauteile einzeln angesteuert und frei bewegt werden. Im Automatikbetrieb werden die fünf genannten Schritte in ihrer Reihenfolge abgearbeitet. Über einen Notschalter lässt sich die Maschine stoppen.

3.4.3 9+1-Spindelschrauber

Mit Hilfe eines 9+1-Spindelschraubers der KSM werden ebenfalls Teile eines Automobilgetriebes automatisch verschraubt. Es werden neun Schrauben senkrecht und eine weitere Schraube waagerecht an einem Bauteil (Getriebe) verschraubt. Das Verschrauben erfolgt ebenfalls in 5 Schritten. Dabei unterscheidet sich in der Hauptsache lediglich der „Verschraubungsschritt“: Eine einfache Schraubeinheit wird waagerecht an das Werkstück geführt und verschraubt dort eine einzelne Schraube. Eine 9-Spindel-Schraubeinheit verschraubt senkrecht von oben das Werkstück. Während die einzelne Schraube bereits im Werkstück eingeschraubt ist, werden für die 9-Spindel-Schraubeinheit Schrauben ähnlich dem vorgenannten Verfahren zugeführt. Dabei verwendet die Schraubenplatte jedoch ein abweichendes Prinzip: statt Mag-

neten verwendet sie Greifer. Des Weiteren verfügt der Werkstückträger über keinen Informationsträger. Für diese Maschine ist ebenfalls ein Bedienpult vorgesehen, das den Wechsel zwischen Automatik- und Handbetrieb ermöglicht.



Abbildung 17: 9+1-Spindelschrauber (Karl Schumacher Maschinenbau GmbH)

4 Ontologien

4.1 Vorstellung des Instruments Ontologien

4.1.1 Grundsätze von Ontologien

Das Konstrukt *Ontologie*, das im Rahmen der Betriebswirtschaftslehre, der Wirtschaftsinformatik und der Forschung zur Künstlichen Intelligenz Verwendung findet, hat zum Ziel, Wissen wiederzuverwenden.[295] Dabei sollen im Bereich des inner- und überbetrieblichen Wissensmanagements divergierende Wissenshintergründe kompensiert oder sogar beseitigt werden.[296]

Das Ziel der Wiederverwendung von Wissen durch Ontologien wird auch deutlich, wenn man sich die Definition für Ontologien von Schreiber, Akkermans et al. (2001), S. 331, vor Augen hält: *Generalized domain schemas are called ontologies*[297], wobei aus Sicht des Software-Engineering ein Domänenschema einem Datenmodell oder Objektmodell ähnelt.[298] Die Verwendung von Ontologien basiert immer auf der Annahme, dass ein Wissensbasiertes System die wahrgenommene Welt von Akteuren reflektiert.[299]

Ontologien bedienen sich bei der Reflektion der wahrgenommenen Welt oft methodischer und architektonischer Eigenheiten.[300] Auf der methodischen Seite benötigen Ontologien ein sehr umfassendes interdisziplinäres Vorgehen bei der Konstruktion. (Analytische) Philosophie und Linguistik bilden hierbei die Basis, um eine „gegebene" – d. h. individuell oder kollektiv wahrgenommene – Realität auf hohem Abstraktionsgrad zu generalisieren. Architektonisch bemerkenswert ist der Umstand, welche zentrale Rolle Ontologien in einem Informationssystem einnehmen können.[301] Guarino schlägt deshalb vor, die Einteilung von Wissensrepräsentationssprachen gemäß Brachman[302] mit den Ebenen logisch, epistemologisch, konzeptuell und linguistisch um die ontologische Ebene zu erweitern, die zwischen der epistemologischen und der konzeptuellen Ebene angesiedelt wird.[303] Die ontologische Ebene wird als die Ebene der Bedeutung angesehen. Hingegen sind die Primitive auf der konzeptuellen Ebene lediglich durch eine kognitive, sprachunabhängige Interpretation bestimmt.

---

295) Vgl. Neches, Fikes et al. (1991), S. 37 ff. Siehe hierzu auch Fn. 125, S. 29.

296) Vgl. Zelewski (2002b), S. 3.

297) Die Generalisierung wird gerade zum Zweck der Wiederverwendung angestrebt.

298) Siehe Kapitel 5.4.3.1, Seite 187, für den CommonKADS-Ansatz von Schreiber, Akkermans et al., dem diese Definition innewohnt.

299) Vgl. Wand, Monarchi et al. (1995), S. 287.

300) Vgl. Guarino (1998), S. 3.

301) Vgl. Alparslan, Dittmann et al. (2002). Ein ontologiebasiertes Informationssystem beinhaltet als Wissensbasiertes System im Kern eine Ontologie als zentrales Element. Siehe hierzu auch Abbildung 21, S. 93.

302) Vgl. Brachman (1979), S. 192. Guarino (1995), S. 10, vernachlässigt dabei die Implementierungsebene, die von Brachman als fünfte Ebene definiert wurde.

303) Vgl. Guarino (1995), S. 10.

In ihrem Streben, das noch junge Forschungsfeld der Ontologien von verwandten Forschungsfeldern extern abzugrenzen und intern einheitlich zu strukturieren, unterscheidet die Forschung zu Ontologien der KI seit kurzer Zeit zwischen leichtgewichtigen (lightweight) und schwergewichtigen (heavyweight) Ontologien.[304]

Leichtgewichtige Ontologien bestehen vornehmlich aus Konzepten, hierarchischen (binären) Relationen zwischen den Konzepten, so genannten *Taxonomien*, und weiteren binären Relationen (die auch Eigenschaften der Konzepte beschreiben sollen). Taxonomien sind Mengen hierarchisch geordneter Klassen.[305] Sie werden besonders zur Klassifikation großer Datenbestände verwendet.[306] Folgt man dieser Erläuterung, so wird es möglich, bspw. das Klassifizierungsschema von Yahoo!© oder einen beliebigen weiteren Indexierungsdienst des WWW als eine Ontologie aufzufassen.[307]

Im Gegensatz zu den leichtgewichtigen Ontologien beinhalten die *schwergewichtigen Ontologien* umfassende Restriktionen hinsichtlich der Semantik. Schwergewichtige Ontologien umfassen die Modellierungsprimitive[308] leichtgewichtiger Ontologien und fügen den leichtgewichtigen Ontologien Regeln (auch: Axiome) und weitere Constraints[309], d. h. Einschränkungen hinsichtlich der Datentypen, Wertebereiche für Variablen usw., hinzu.

Mit Hilfe dieser Regeln wird eine formale Semantik ermöglicht. Eine formale Darstellung wird für die Verarbeitung innerhalb eines Wissensbasierten Systems vorausgesetzt.[310] Wissensbasierte Systeme werden durch schwergewichtige Ontologien in die Lage versetzt, neben

304) Vgl. bspw. Gómez-Pérez, Fernández-López et al. (2004), S. 8; Fluit, Sabou et al. (2004), S. 415 f.

305) In dieser Arbeit werden die Begriffe Klassen und Konzepte synonym verwendet, weil von sprachanalytischen Feinheiten abgesehen werden kann. Für eine ausführliche Darstellung der Zusammenhänge siehe Zelewski (2005), S. 149 f.

306) Taxonomien werden in allen Repräsentationssprachen für Ontologien ermöglicht und können deshalb als integraler Bestandteil (jedoch nicht unbedingt notwendiger) einer Ontologie angesehen werden. Artverwandt mit Taxonomien sind *Identifikationssysteme*, die statt Konzepten individuelle Identifikatoren umfassen (vgl. Strang (2003), S. 2).
Beispiele für Taxonomien sind die Klassifikation von Wirtschaftszweigen des Statistischen Bundesamts (vgl. http://www.destatis.de/allg/d/klassif/wz2003.htm, Zugriff am 18.03.2007) und der United Nations Standard Products and Services Code® (UNSPSC; vgl. http://www.unspsc.org, Zugriff am 18.03.2007).
Beispiele für Identifikationssysteme sind die D-U-N-S Nummer zur Identifikation von Unternehmen (vgl. http://dbgermany.dnb.com/German/default.htm?Loc=/German/Database/duns.htm, Zugriff am 18.03.2007) und Telefonnummern (vgl. http://www.telefonbuch.de, Zugriff am 18.03.2007).

307) Für das Beispiel einer Untersuchung von Yahoo!© auf dessen Ontologieentwicklungsmöglichkeit siehe Labrou, Finin (1999).

308) Als Modellierungsprimitive werden die elementaren Bausteine, die vor der Modellierung bereits eine festgelegte Bedeutung (im Sinne einer vorgesehenen Verwendungsweise) besitzen und für die Modellierung notwendig sind, bezeichnet. Bspw. können Buchstaben als Modellierungsprimitive aufgefasst werden, mit denen Wörter oder Sätze gebildet werden können.
Zu den Modellierungsprimitiven für leichtgewichtige Ontologien werden an dieser Stelle *Konzepte* und *Relationen* gezählt.

309) Im Folgenden umschließt der Begriff „Regeln" auch die genannten Constraints, weil diese im eigentlichen Sinne auch als Meta-Regeln, die eine Verwendungsweise von möglichen Konzepten „regeln", aufgefasst werden können.
Siehe zum Themenbereich *Regeln* auch Kapitel 2.1.2.3, S. 24 ff., und insbesondere die Fn. 105, S. 26.

310) Siehe hierzu auch Kapitel 2.1.2.2, S. 23 f.

der syntaktisch korrekten Verwendung von Konzepten auch eine semantisch korrekte – d. h. bedeutungsgerechte – Verwendung von Konzepten zu verarbeiten.

Gemäß der Definition von Kapitel 2.1.2.4, S. 28 ff., werden in dieser Arbeit nur schwergewichtige Ontologien als Ontologien aufgefasst, weil es als konstituierendes Merkmal angesehen wird, dass eine Ontologie über Regeln verfügt.[311] Das heißt, wenn im Text von „Ontologien" die Rede ist, dann sind damit schwergewichtige Ontologien gemeint. Auch um sich gegenüber dem Forschungsfeld der Konzeptuellen Modellierung[312] und seiner Repräsentationssprachen besser abzugrenzen[313], wird dieser Auffassung gefolgt. In den folgenden Ausführungen wird deshalb auf diesen Umstand nicht mehr gesondert hingewiesen und lediglich der Begriff Ontologien gemäß der Definition, die dieser Arbeit zugrunde liegt, angewendet.

Insbesondere drei Charakteristika lassen sich für die Darstellung der Bedeutung von Ontologien heranziehen. Diese lassen sich auch zur Abgrenzung von Ontologien gegenüber weiteren Instrumenten zur Wissensrepräsentation[314] nennen:[315]

- Kommunikation,
- automatisches Schließen (Reasoning) und
- Repräsentation und Wiederverwendung von Wissen.

4.1.2 Arten von Ontologien

Obwohl der grundsätzliche Aufbau von Ontologien in den meisten Fällen ähnlich ist, weil sie einer gemeinsamen Definition folgen, lassen sie sich in verschiedene Kategorien unterteilen.

311) Bspw. bezweifelt Kraus, dass leichtgewichtige Ontologien überhaupt Ontologien sind. Nach seiner Ansicht ist eine formale Semantik, die leichtgewichtigen Ontologien fehlt, eine Grundvoraussetzung für eine Ontologie (vgl. Kraus (2003), S. 41).

312) Vgl. zu konzeptuellen Modellen und zur konzeptuellen Modellierung Frank (1998), S. 9 ff.; Frank (2000), S. 709 ff.; Mylopoulos (1998), S. 130, und umfassend Boman, Bubenko et al. (1997).

313) Die Abgrenzung erfolgt hierbei nicht im Sinne eines Abschlusses gegenüber der Konzeptuellen Modellierung, sondern vielmehr in Form einer inhaltlichen Präzisierung des Konstrukts „Ontologie" gegenüber einem konzeptuellen Modell. Bei der Konzeptuellen Modellierung liegt der Schwerpunkt oftmals auf dem Explikationsvorgang (Explikation des Wissens von Akteuren für die Nutzbarmachung mittels Wissensbasierter Systeme), während der Schwerpunkt bei Ontologien zusätzlich auf der formalen Wissenswiederverwendung liegt.

314) Siehe hierzu Kapitel 4.1.3, S. 79 ff.

315) Ursprünglich wurden diese Charakteristika als *Haupteinsatzgebiete von Ontologien* von Strang (vgl. Strang (2003), S. 3) aus dem Werk von Uschold und Grüninger (Uschold, Grüninger (1996)) destilliert. Uschold und Grüninger sprechen dabei im Original von den drei *Hauptkategorien* für den Einsatz von Ontologien (vgl. Uschold, Grüninger (1996), S. 13, Abbildung 1). Während Uschold und Grüninger die Förderung der Kommunikation neben der Ermöglichung der Interoperabilität zwischen Systemen und die Unterstützung von Systems-Engineering-Aufgaben noch als einen eigenständigen Punkt berücksichtigen, werden das Automatische Schließen sowie die Repräsentation und Wiederverwendung von Wissen nur mittelbar genannt. Das Automatische Schließen dient Uschold und Grüninger insbesondere zur Überprüfung von Konsistenz (vgl. Uschold, Grüninger (1996), S. 45). Repräsentation und Wiederverwendung von Wissen werden für die Zwecke der Erfüllung von Systems-Engineering-Aufgaben (vgl. Uschold, Grüninger (1996), S. 12 ff. neben Aufgaben der Spezifikation) genannt.

Je nach Forschungsansatz werden Einteilungen in Abhängigkeit vom Wiederverwertungsstandpunkt[316], von der Allgemeingültigkeit (Generalität)[317] der Ontologien und von der Größe und Art ihrer Struktur[318] vorgenommen. In dieser Arbeit wird eine Art der Kategorisierung aufgegriffen, die Ontologien nach ihrer Allgemeinheit (Generizität) und ihrem Anwendungsgebiet (Domäne) unterscheidet, d. h., es wird nach dem Fokus der Konzeptualisierung unterschieden.[319] Es wird mit zunehmender Allgemeinheit unterschieden in:[320]

- Domänen-Ontologien,
- Commonsense-Ontologien,
- Methoden-Ontologien,
- Aufgaben-Ontologien und
- Repräsentations-Ontologien.

---

316) Vgl. Mizoguchi, Ikeda (1996), S. 5. Hier werden Ontologien in *Arbeitsplatz-, Aufgaben-, Domänen-* und *allgemeine Ontologien* unterschieden. Von Mizoguchi und Ikeda wird dabei unterstellt, dass es bei Arbeitsplatz-Ontologien gegenüber allgemeinen Ontologien tendenziell weniger häufig zu einer Wissenwiederverwendung – d. h. einem weiteren Einsatz einer bestehenden Ontologie – kommt.

317) Vgl. Guarino (1998), S. 7 ff. Guarino unterscheidet dabei zwischen *feineren Ontologien* (detailed ontologies) und *gröberen Ontologien* (coarse ontologies), als Beispiele nennt er hierzu Referenz-Ontologien (reference ontologies) bzw. verteilbare Ontologien (sharable ontologies). Für gröbere Ontologien lassen sich tendenziell aufgrund ihrer erhöhten Generizität mehr Akteure finden, die diese Ontologie als eine gemeinsame Basis akzeptieren und verwenden (höhere Allgemeingültigkeit) als für eine spezialisiertere (weil feinere) Ontologie.
An dieser Stelle wird deutlich, dass die genannten Einteilungen (Wiederverwendbarkeit und Generalität) nicht disjunkt zueiander stehen, denn die breitere Akzeptanz impliziert auch eine erhöhte Wiederverwendung von Ontologien. Weil jedoch die genannten Ansätze für Einteilungen nur der Verdeutlichung dienen sollen, wird an dieser Stelle nicht weiter auf die dahinterstehende Kategorisierungsproblematik eingegangen.

318) Vgl. van Heijst, Schreiber et al. (1997), S. 192 f. Die Autoren unterscheiden in: *Terminologie-Ontologien* (terminological ontologies), *Informations-Ontologien* (information ontologies) und *Wissensmodellierungs-Ontologien* (knowledge modelling ontologies).

319) Weitere Unterteilungen werden in Vanwelkenhuysen, Mizoguchi (1995), S. 4.3 f., und Gómez-Pérez, Fernández-López et al. (2002), S. 16, vorgestellt. Die hier übernommene Unterteilung ist in der Literatur häufig zu finden, vgl. z. B. Zelewski, Schütte et al. (2001), S. 194 f.; Studer, Benjamins et al. (1998), S. 186; Studer Fensel et al. (1999), S. 5; Falasconi, Stefanelli (1994), S. 82 ff. Problematisch dabei ist die nicht immer disjunkte Verwendung der Mengenbegriffe; so kann unter Umständen eine Commonsense-Ontologie auch als Domänen-Ontologie aufgefasst werden, nämlich für die Domäne „Weltwissen". Für die Problemstellung dieser Arbeit ist jedoch die Unterteilung ausreichend, um zum Ziel der FMEA-Ontologieentwicklung zu führen; denn sie stellt einen Kompromiss zwischen Klarheit und Überschaubarkeit der Kategorisierung einerseits und ihrer Genauigkeit andererseits dar.

320) In der vorliegenden Aufstellung kann eine spezifische Domäne als Basis für das Attribut „Allgemeinheit" betrachtet werden. Eine Commonsense-Ontologie umschließt mehrere Domänen. Methoden- und Aufgaben-Ontologien beinhalten das Wissen, um aufbauend spezifische Domänen darzustellen. Hingegen sind Repräsentations-Ontologien als domänenunabhängig und damit mit dem Attribut der höchsten Stufe der Allgemeinheit anzusehen. Vgl. hierzu auch Studer, Benjamins et al. (1998), S. 186.

4.1.2.1 Domänen-Ontologien

In diesen Ontologien werden Konzepte innerhalb einer Domäne und ihre Beziehungen sowie Regeln, die für die Domäne gültig sind, bereitgestellt. Sie legen damit Einschränkungen für die formale Semantik von Domänenwissen fest.[321] Eine Domäne wird dabei als ein darzustellender Ausschnitt der Welt angesehen.[322] Die *Enterprise Ontology*[323] bspw. stellt eine Domänen-Ontologie dar, die für gewerbliche Unternehmen relevante Begriffe umfasst. Auch die Kompetenz-Ontologie für die DMT GmbH, die in dieser Arbeit vorgestellt wird[324], ist als Domänen-Ontologie für den Bereich Technologie-Dienstleistung mit einem weiteren Schwerpunkt auf dem Bereich des Kompetenzmanagements zu sehen.

4.1.2.2 Commonsense-Ontologien

Commonsense-Ontologien, auch als „allgemeine“ oder „generische“ Ontologien bezeichnet, umfassen allgemeine Konzepte, die über mehrere Themengebiete hinweg – im Gegensatz zu bspw. Domänen-Ontologien – Gültigkeit besitzen. Mit ihnen wird versucht, „allgemeines Weltwissen“ zu erfassen[325] und durch Konzepte wie Zeit, Raum, Zustand, Ereignis, Prozess, Aktion, Komponente usw. darzustellen.[326] Beispiele für bereits entwickelte Commonsense-Ontologien sind etwa die *Suggested Upper Merged Ontology (SUMO)*[327], das *Generalized Upper Model (GUM)*[328], *Cyc*[329] und *SENSUS*[330].

4.1.2.3 Methoden-Ontologien

Durch Methoden-Ontologien werden Konzepte, Relationen und Regeln für (systematische) Verfahren zur Gewinnung von Lösungen für Klassen gleichartiger Probleme definiert[331]; sie legen die Primitive fest, mit denen die Operationen einer Methode beschrieben werden können. Um die Wiederverwendung einer Methoden-Ontologie zu ermöglichen, werden die Methoden durch abstrakte und domänenunabhängige Konzepte dargestellt. In der *Propose-and-Revise-Method-Ontology* („Vorschlagen-und-überprüfen-Ontologie“) des Protégé-II-Projekts

---

321) Vgl. van Heijst, Schreiber et al. (1997), S. 193.

322) Vgl. Owsnicki-Klewe, von Luck et al. (2003), S. 157.

323) Siehe Fn. 515, S. 116.

324) Siehe Kapitel 4.4.3, S. 118 f.

325) Vgl. Zelewski Schütte et al. (2001), S. 195.

326) Vgl. Puppe, Stoyan et al. (2003), S. 623.

327) Die Suggested Upper Merged Ontology (SUMO) ist aus Bemühungen einer IEEE-Arbeitsgruppe (IEEE Standard Upper Ontology) entstanden, die die Entwicklung einer allgemeinen, erweiterbaren und standardisierbaren Ontologie zum Ziel haben; für eine kurze Darstellung vgl. Pease, Niles et al. (2002) sowie http://ontology.teknowledge.com/, Zugriff am 18.02.2004.

328) Vgl. Bateman, Magnini et al. (1994).

329) Zu Cyc siehe Kapitel 4.4.1, S. 116 f.

330) Zu SENSUS siehe Kapitel 4.4.4, S. 119 f.

331) Vgl. Studer, Fensel et al. (1999), S. 5.

werden bspw. Nebenbedingungen für Methoden und Statusvariablen wie Ein- und Ausgaben der Methoden definiert.[332]

4.1.2.4 Aufgaben-Ontologien

Aufgaben-Ontologien umfassen Spezifikationen allgemeiner Aufgabentypen, die in unterschiedlichen Domänen vorkommen können. Sie beinhalten Konzepte, Relationen und Regeln generischer Aufgabenklassen[333], wie zum Beispiel eine Ontologie für parametrische Design-Aufgaben.[334] Da sich sowohl Aufgaben- als auch Methoden-Ontologien auf die Abbildung sehr begrenzter (Realitäts-)Ausschnitte konzentrieren, weisen sie in dieser Hinsicht eine starke Ähnlichkeit mit Domänen-Ontologien auf. Die Aufgaben- und Methoden-Ontologien ermöglichen durch ihre Fokussierung auf Aufgaben bzw. Methoden eine schlussfolgernde Sicht auf Domänenwissen.[335] Unter Verwendung von Aufgaben- und Methoden-Ontologien lässt sich domänenspezifisches Wissen in Form von Aufgaben und Methoden als Instanzen repräsentieren, dabei wirken die Modellierungsprimitive wie eine „Brille“, die eine problemorientierte Sicht auf eine Domäne ermöglicht.

4.1.2.5 Repräsentationsontologien

Um die Ausdrucksmittel von Repräsentations- oder Modellierungssprachen zu spezifizieren, können Repräsentations-Ontologien eingesetzt werden.[336] Sie beinhalten Konzeptualisierungen von Wissensrepräsentationssprachen. Im Gegensatz zu den anderen Arten von Ontologien stellen Repräsentationsontologien keine Annahmen darüber auf, was (welcher Realitätsausschnitt) in einer Ontologie oder einem Modell abgebildet wird, sondern nur darüber, wie das Wissen repräsentiert wird.[337] Ein bekanntes Beispiel einer Repräsentationsontologie ist die Frame-Ontology.[338]

4.1.3 Abgrenzung von anderen Instrumenten

4.1.3.1 Ansätze zur Repräsentation von Wissen

Ontologien als spezielle Form *konzeptueller Modelle* sind das Ergebnis eines Explikationsvorgangs. In einem konzeptuellen Modell sollen die Wissenshintergründe der Akteure sprach-

---

332) Vgl. Rothenfluh, Gennari et al. (1994), S. 43.5 f.

333) Vgl. Fensel, Motta et al. (1997), S. 115.

334) Vgl. Fensel, Motta et al. (1997), S. 115 ff., und Motta, Zdrahal (1998), S. 442 ff. Unter parametrischem Design versteht man eine Klasse von Gestaltungsproblemen, die die Zuweisung von Werten zu einer Menge von Design-Parametern erfordern (vgl. Motta, Zdrahal (1998), S. 440).

335) Fensel (2001), S. 12, im Original: „Task and method ontologies provide a reasoning point of view on domain knowledge“.

336) Alternativ kann auch von Meta-Ontologien gesprochen werden (vgl. Zelewski, Schütte et al. (2001), S. 195).

337) Vgl. Studer, Fensel et al. (1999), S. 5.

338) Zur Frame-Ontology siehe Kapitel 4.4.2, S. 117 f.

lich expliziert werden, um dieses Hintergrundwissen bspw. dem Zugriff Wissensbasierter Systeme zugänglich machen zu können.[339]

In der Forschung lassen sich zahlreiche Ansätze finden, die sich mit der computergestützten Repräsentation und automatischen Auswertung von Wissen auseinander setzen. Insbesondere das Forschungsgebiet der Künstlichen Intelligenz hat hierzu zahlreiche Ansätze geliefert. Je nach Anwendungshintergrund wurden zum Teil recht unterschiedliche Ansätze zur Repräsentation von Wissen entwickelt. Eine Auswahl von Repräsentationsmöglichkeiten für Wissen umfasst:

- Thesauri[340],
- Topic Maps[341],
- Semantische Netze[342],
- Conceptual Graphs[343],
- Frames[344] und
- Entity-Relationship-Modelle.[345]

Um ein tieferes Verständnis für das Instrument *Ontologien* zu ermöglichen, werden die genannten möglichen Instrumente zur Repräsentation von Wissen kurz vorgestellt, und es wird versucht, diese gegenüber Ontologien abzugrenzen.

Unter Vorgriff auf die folgenden Ausführungen lässt sich feststellen, dass Ontologien die unterschiedlichen Charakteristika der vorgestellten Ansätze zusammenfassen. Denn Ontologien ermöglichen Ableitungen (wie semantische Netze), verwenden Klassifizierungen (wie Taxonomien), beschreiben Konzepte, auf die sich eine Benutzergruppe geeinigt hat (wie Thesauri), bilden einen Realitätsausschnitt ab (wie ERM und Frames), und sie dienen der Navigation in Wissensbeständen (wie Topic Maps).[346]

339) Siehe hierzu auch Fn. 313, S. 76.

340) Vgl. DIN 1463-1:1987.

341) Vgl. Park, Hunting (2003); ISO/IEC 13250:2002.

342) Vgl. Strube, Habel et al. (2003), S. 40 ff.

343) Vgl. Sowa (2000), S. 476 ff.

344) Vgl. Minsky (1975).

345) Vgl. Chen (1976).

346 Vgl. Staab (2002), S. 201.

4.1.3.2 Ausgewählte weitere Ansätze zur Repräsentation von Wissen

4.1.3.2.1 Thesauri

Thesauri wurden ursprünglich in der Bibliographie entwickelt. Ein Thesaurus im Bereich der Information und Dokumentation ist eine geordnete Zusammenstellung von Konzepten und ihren (vorwiegend natürlichsprachlichen) Bezeichnungen, die in einem Dokumentationsgebiet zum Indexieren, Speichern und Wiederauffinden dient.[347] Ein Thesaurus umfasst eine Menge vorgegebener Relationen, mit deren Hilfe ein Interessengebiet strukturiert wird.[348] Die festgelegten Relationen können nicht erweitert werden, i. d. R. werden Relationen berücksichtigt wie: `aehnlich_wie`, `unterbegriff_von` und `synonym_zu`.

Mit Hilfe der festgelegten Relationen können Konzepte und eingeschränkt das Wissen über ihre Relationen zueinander repräsentiert werden. Aufgrund der beschränkten Anzahl feststehender Relationen ergibt sich kaum eine Nutzung des Wissens zur Kommunikation und zur Wiederverwendung des Wissens. Automatisches Schließen wird nicht unterstützt. In der Regel umfasst die Mächtigkeit der Ontologierepräsentationssprachen die Möglichkeiten der Thesauri-Entwicklung vollständig.

4.1.3.2.2 Topic Maps

Topic Maps ermöglichen unterschiedliche gleichzeitige Sichten auf Mengen von Konzepten (Informationsobjekte).[349] Sie bestehen primär aus den Konstrukten: *Topik* (topic)[350], *Vorkommen* von Topiken (topic occurrence), *Assoziationen* (associations), *Namen* und *Gültigkeitsbereichen* (scope).[351] Topic Maps werden durch den zurzeit gültigen Standard ISO/IEC 13250:2002 einheitlich definiert.

Ein Wissensbereich kann durch eine Topic Map erschlossen werden. Zunächst werden dazu alle Topiken definiert, anschließend die erforderlichen Assoziationen formuliert.[352] Eine Topik verfügt dabei in der Regel über eine Bezeichnung (Name), es ist jedoch auch zulässig, dass keine Bezeichnung oder mehrere Bezeichnungen angegeben werden.[353] Anschließend werden die Topiken mit den Assoziationen verbunden. Alle Topiken werden über Assoziatio-

347) DIN 1463-1:1987, S. 2.

348) Beispielhaft für einen Thesaurus sei der Thesaurus für Technik und Management (vgl. Fachinformationszentrum Technik (2000)) genannt.

349) Vgl. ISO/IEC 13250:2002, S. 1.

350) Auch als *Knoten* bezeichnet (vgl. Strang, S. 2). Diese entsprechen den spezifischen Konzepten, die mit einer Topic Map verortet werden sollen.

351) Vgl. Biezunski (2003), S. 18 ff.

352) Eine Assoziation entspricht einer zweistelligen Relation.

353) Vgl. Biezunski (2003), S. 19.

nen miteinander zu einer Abbildung (Map) verknüpft. Assoziationen besitzen einen Typ, mit demselben Typ verknüpfte Topiken besetzen eine definierte Rolle in der Assoziation.[354]

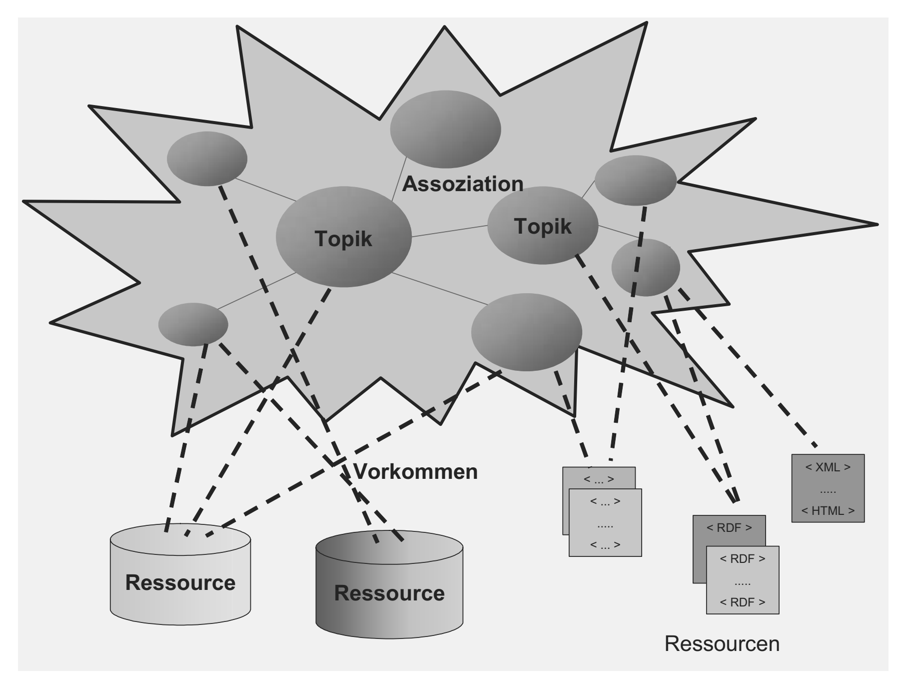


Abbildung 18: Grundaufbau von Topic Maps

Gültigkeitsbereiche schränken die Gültigkeit von Assoziationen ein, definieren den Kontext einer Topik und dienen der Herstellung unterschiedlicher Sichten. Das Vorkommen von Topiken in der „realen“ Welt (topic occurrences) bspw. als Dokument wird als Verbindung, in der Abbildung 18 mittels einer gestrichelten Kante, von Topik und Informationsressource (hier: Dokument) dargestellt.

Abbildung 18 verdeutlicht den Grundaufbau von Topic Maps. Die „Wolke“ stellt hierbei die eigentliche Topic Map mit ihren Hauptelementen dar. Eine Topic Map kann als Navigationsinstrument auf die real vorhandenen Inhalte (Objekte), die Vorkommen (occurrences) der Topiken, verwendet werden. Dem Nutzer präsentiert sich dabei die Navigation als eine Ebene über den eigentlichen Objekten, die viele verschiedene, aber immer sinnvolle Wege durch die Objekte anbietet. Jeder Nutzer kann so seinen – den individuellen Präferenzen entsprechenden – Weg durch ein Themenfeld wählen.

Topic Maps können zur Repräsentation und Kommunikation von Wissen herangezogen werden. Vornehmlich dienen sie zur Generierung einer Meta-Ebene, die die Navigation über ei-

354) Als anschauliches Beispiel für eine Assoziation seien die Topiken "Newton" und "Kraft" gegeben. Diese sind verknüpft mit der Assoziation "ist die Einheit von". "Newton" besetzt dabei die Rolle "Einheit" und "Kraft" besetzt dabei die Rolle "Größe".

nen Wissensbestand (Objekte) erlaubt. Die Berücksichtigung von Regeln, die insbesondere non-deduktive Schlussfolgerungen erlauben, wird nicht ermöglicht. Topic Maps sind damit weniger ausdrucksstark als Ontologien, weil sie ein automatisches Schließen nicht erlauben.

4.1.3.2.3 Semantische Netze

Ein *Semantisches Netz*[355] ist ein Gedächtnismodell[356], das aus einer definierten Menge von Konzepten (begriffliche Entitäten) und den zwischen diesen existierenden Relationen besteht.[357] Es verfügt über lediglich zwei Arten von formalen Elementen, die *Knoten* und die gerichteten *Kanten*. Mit Hilfe der Knoten werden Objekte (Konzepte) dargestellt. Die Kanten repräsentieren die Beziehungen zwischen diesen Objekten. Abhängig vom Anwendungsgebiet werden verschiedenartige Beziehungen dargestellt. Besonders hervorzuheben sind jedoch die `ist_ein`- und die `hat_`-Beziehung, die eine Spezialisierungs- oder eine Elementbeziehung ausdrücken.[358] Aufgrund der netzartigen Struktur der Gedächtnismodelle werden diese als Semantische Netze bezeichnet. Semantische Netze und prädikatenlogische Formeln repräsentieren die gleiche Information in unterschiedlichem Format (Knoten entsprechen Konzepten und gerichtete Kanten entsprechen zweistelligen Prädikaten).[359] Der Nachteil Semantischer Netze liegt in der umfangreichen und unübersichtlichen Darstellung komplexerer Sachverhalte. Dieses Problem verspricht der Ansatz der *Frames* im übernächsten Unterkapitel zu lösen.

Semantische Netze können zur Repräsentation und Kommunikation von Wissen herangezogen werden. Hauptunterschiede gegenüber Ontologien liegen in der fehlenden Darstellungsmöglichkeit von Regeln und der hiermit verbundenen Möglichkeit des automatischen Schließens sowie der fehlenden Möglichkeiten der Vererbung von Eigenschaften auf untergeordnete Konzepte. Ontologien stellen damit eine Fortentwicklung Semantischer Netze dar.[360]

4.1.3.2.4 Conceptual Graphs

Der Ansatz der konzeptuellen Graphen (Conceptual Graphs (CGs))[361] ist eine Kombination Peirce'scher Logik mit Semantischen Netzen.[362] Ein konzeptueller Graph ist ein bipartiter

355) Bibel (1993) spricht in diesem Zusammenhang auch von assoziativen Netzen. Semantische Netze sind nicht zu verwechseln mit dem Semantic Web nach Berners-Lee, Hendler et al. (2001).

356) Vgl. Strube, Habel et al. (2003), S. 40.

357) Vgl. Haun (2000) S. 66.

358) Vgl. Silberbusch (1990), S. 167.

359) Die Aussage „gleich" trifft streng genommen nur zu, wenn die Prädikatenlogik auf zweistellige Prädikate reduziert wird.

360) Vgl. Büchel (2002), S. 36.

361) Vgl. zur Verwendung von CG Sowa (1984).

362) Vgl. Sowa (2000), S. 23. C. S. Peirce übernahm die Ansätze von G. Boole, der Aussagen Wahrheitswerte (wahr, falsch) zuordnete und die logischen Operatoren UND, ODER und NICHT verwendete, und erweiterte diese um Wahrheitstafeln (truth tables) und Implikation (material implication). Ferner modifizierte er das Boole'sche *exklusive* ODER zu einem *inklusiven* ODER.

Graph mit zwei Arten von Knoten.[363] In einem konzeptuellen Graphen repräsentieren viereckige Objekte Konzepte, und Kreise repräsentieren Relationen zwischen Konzepten. Jedes Viereck beherbergt auf der linken Seite ein Typenfeld (type field), das mit einem Typ-Etikett (type label) versehen wird. Die rechte Seite bildet das Referenzfeld, das einen Namen, einen Quantor oder eine Kardinalität wiedergibt. Die Konzeptrelation gibt an, wie zwei Referenzen eines Konzepts miteinander verbunden werden.

Die Semantik von konzeptuellen Graphen entspricht der Ausdrucksstärke der Prädikatenlogik erster Stufe, deshalb können Ontologien mit der Repräsentationsmöglichkeit konzeptueller Graphen erstellt werden. Das Erstellen von Regeln als Voraussetzung für ein automatisches Schließen als Teil einer Ontologie wird jedoch nur insofern unterstützt, als dass die (computerverarbeitbare, formale) Repräsentation von komplexen Regeln in ihrer Darstellung kaum von der üblichen Darstellung der Prädikatenlogik erster Stufe zu unterscheiden ist. Damit der ursprüngliche Vorteil der intuitiven (weil einfachen graphischen) Darstellung gegenüber der herkömmlichen Darstellung in Prädikatenlogik voll zum Tragen kommen kann, lassen sich lediglich leichtgewichtige Ontologien mit CGs entwickeln.[364]

4.1.3.2.5 Frames

Minsky nennt als Quintessenz seines *Frame*-Konzepts: Wenn jemand auf eine neue Situation stößt (oder jemand einen substantiellen Sichtwechsel auf ein gegenwärtiges Problem vollzieht), so selektiert er aus seinem Gedächtnis eine Struktur, die als *Frame* bezeichnet wird. Es handelt sich hierbei um einen Rahmen (Frame) aus der Erinnerung, der durch die Veränderung so vieler Details wie nötig adaptiert wird, um der Realität zu entsprechen.[365]

Haun definiert einen Frame als eine strukturierte Repräsentation einer Entität oder einer Klasse von Entitäten im Sinne einer Verallgemeinerung semantischer Netzwerkmodelle.[366] Frames tragen dem Bedürfnis Wissensbasierter Systeme nach einer einheitlich strukturierten Wissensbasis Rechnung, um die Maschinenverarbeitbarkeit in für den Benutzer angemessener Zeit zu gewährleisten. Frames enthalten eine definierte Anzahl mit speziellen Instanzen oder Daten gefüllter Slots. Slots können auch mit Default-Werten und Verweisen auf andere Frames gefüllt werden. Ein gefüllter Slot in diesem Sinne ist eine vom Benutzer definierte Datenstruktur einschließlich der Operationen, die auf diesen Daten ausgeführt werden.[367] Frames sind sehr ähnlich den Objekten aus der objektorientierten Programmierung (OOP). Die Abbildung 19 zeigt den Frame eines Dreiecks.

363) Vgl. Sowa (2000), S. 477.

364) Hierzu definiert Sowa eine Wissensbasis als einen informellen Begriff für eine Sammlung von Wissen, zu der als eine Komponente eine Ontologie gehört. Neben der Ontologie kann eine Wissensbasis weitere Komponenten enthalten, die in einer deklarativen Sprache, etwa als Expertensystemregeln, repräsentiert werden (vgl. Sowa (2000), S. 495).

365) Vgl. Minsky (1975), S. 212.

366) Vgl. Haun (2000), S. 103.

367) Vgl. Silberbusch (1990), S. 172.

Frame	Dreieck
Unterframe_von	Polygon
Anzahl_Eckpunkte	drei
Anzahl_Kanten	drei
gleichschenkelig	ja/nein
Farbe	unbekannt

Abbildung 19: Frame eines Dreiecks[368]

Weil Slots die charakteristischen Eigenschaften der modellierten Objekte beschreiben, unterscheiden sich Frames von Semantischen Netzen. Bei letzteren werden die Knoten nicht weiter unterteilt. Des Weiteren sind Frames auch in Netzen organisiert, sie weisen hierbei jedoch eine Vererbungshierarchie (bspw. mittels is_a- oder Unterframe_von-Relation) auf, die bei Semantischen Netzen nicht vorkommt.

Frames können zur eingeschränkten Repräsentation und Kommunikation von Wissen herangezogen werden. Der Ansatz der Frames allein reicht nicht aus, um Ontologien zu repräsentieren (aufgrund der fehlenden Möglichkeit zur direkten Darstellung von Regeln)[369]. Die Vorteile dieser Modellierung werden jedoch in einigen Repräsentationssprachen für Ontologien genutzt (bspw. F-Logic).

4.1.3.2.6 Entity-Relationship-Modelle

Das auf Chen[370] zurückgehende Entity-Relationship-Modell (ERM) unterscheidet hauptsächlich in Entitäten (Entities) und Relationen (Relationships). Als Entität wird ein „Ding“ verstanden, das sich deutlich identifizieren lässt, z. B. eine Person, Unternehmen oder Ereignis. Eine Relation charakterisiert die Verbindung zwischen Entitäten (zum Beispiel kann Vater-Sohn eine Relation zwischen zwei Person-Entitäten darstellen). Gleichartige Entitäten werden zu Entitätstypen (auch Klassen) zusammengefasst und gleichartige Relationen zu Relationstypen (Relationship-Type). Durch Attribute werden Entitäten beschrieben. Ein Schlüsselattribut oder mehrere Schlüsselattribute ermöglichen die eindeutige Identifikation von Entitäten. Kanten verbinden die unterschiedlichen Informationsobjekte (Entitäten, Entitätstypen, Attribute, Relationen, Relationstypen) des ERM. Zur Kennzeichnung der Relationstypen zwischen

368) Entnommen aus Haun (2000), S. 104.

369) Das Konzept *Frames* unterstützt lediglich den Existenzquantor und die Konjunktion aus der Prädikatenlogik. Aufgrund fehlender Negation und Disjunktion können keine Implikationen dargestellt werden. Hieraus folgt, dass keine Regeln für ein automatisches Schließen dargestellt werden können. Vgl. hierzu auch Bibel (1993), S. 46, und Sowa (2000), S. 147 ff.

370) Vgl. Chen (1976).

Entitätstypen tragen die Kanten Kardinalitäten.[371] Grundsätzlich unterscheidet man 1:1, 1:n und m:n Beziehungen. Die Abbildung 20 zeigt die Darstellung eines einfachen ERM in Anlehnung an Chen.[372]

Es lässt sich in einem ERM nicht immer jeder Sachverhalt eindeutig darstellen, und es ist durchaus denkbar, dass eine Relationship auf einer anderen Aggregationsstufe[373] bspw. zu einem Entity wird.

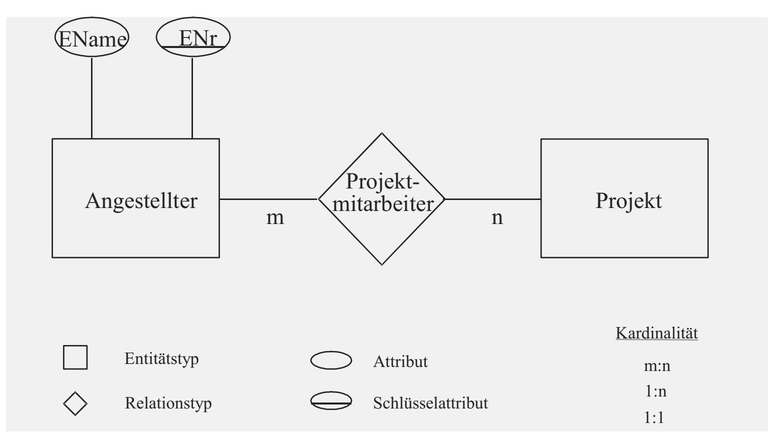


Abbildung 20: Einfaches Entity-Relationship-Modell[374]

Die Ausdrucksmächtigkeit von Entity-Relationship-Modellen reicht lediglich aus, um leichtgewichtige Ontologien darzustellen, weil nicht die Möglichkeit besteht, Regeln zu formulieren oder Vererbungshierarchien darzustellen. Ihr Beitrag zur Repräsentation und zur Kommunikation von Wissen bleibt entsprechend schwach.

4.1.4 Anwendung von Ontologien

4.1.4.1 Durchführung der Entwicklung von Ontologien

Bei der Entwicklung von Ontologien ergeben sich einige implizite Probleme, so sind die meisten Definitionen zu Ontologien unbestimmt sowohl in Bezug auf die Beschaffenheit der unterstellten Realität als auch auf die Erkennbarkeit realer Phänomene. Die Literatur hierzu erweckt oftmals den Eindruck, Ontologien spiegelten die Welt wider, so dass der philosophi-

371) Vgl. Schütte (1998), S. 94.

372) Vgl. Chen (1976), S. 19.

373) Zur Problematik der Aggregation siehe auch Schütte (1998), S. 98.

374) In Anlehnung an Chen (1976), S. 19.

sche Terminus technicus „Ontologie“ zu Recht genutzt werden könnte. Die pluralische Redeweise von Ontologien deutet jedoch schon darauf hin, dass es mehrere „Welten“ gibt.[375]

Unter der Bedingung einer naiv-realistischen Sichtweise, nach der eine Erfahrbarkeit von Realität „an sich“ – unabhängig von sensorisch oder kognitiv bedingten Verzerrungen des erkennenden Subjekts – möglich ist[376], fallen Ontologie (als philosophische Disziplin) und Erkenntnistheorie als analog zusammen, so dass eine singuläre Rede zulässig wäre. Modernere Erkenntnispositionen, wie z. B. der Kritische Realismus Albertscher Prägung[377], der hypothetische Realismus eines evolutionären Erkenntnisprogramms im Sinne Vollmers[378] oder, aus einer anderen Denkrichtung – dem Methodischen Konstruktivismus[379] oder Methodischen Kulturalismus[380] – kommend, ein gemäßigter Konstruktivismus[381] betonen jedoch die aktive und konstruktive Leistung eines erkennenden Subjekts.

Im Folgenden werden die Elemente *Domäne* und *Konzeptualisierung*, die bei der Entwicklung von Ontologien zu bestimmen sind, hinsichtlich ihrer erkenntnistheoretischen Implikationen diskutiert, um die Schwierigkeiten, die mit der Entwicklung von Ontologien verbunden sind, zu verdeutlichen.

Eine Domäne[382] (oder reales System)[383] wird häufig als ein Ausschnitt der Realität verstanden, der unabhängig zu einem modellierenden Subjekt als gegeben postuliert wird.[384] Diese Sichtweise ist zwar aus der Perspektive eines naiven Realismus konsequent, nicht jedoch akzeptabel aus dem Blickwinkel einer aufgeklärten erkenntnistheoretischen Position. Wird sich bspw. auf eine erkenntnistheoretische Position des gemäßigten Konstruktivismus bezogen, würde bereits die Domäne als eine von Subjekten konzeptualisierte Entität begriffen. Somit bedeutet die Domäne selbst bereits das Resultat einer Vor-strukturierung des Gegenstandsbereichs.[385]

Noch deutlicher als bei der Domäne tritt die Zweck- und Erkenntnisabhängigkeit bei der Konzeptualisierung hervor. Unter einer *Konzeptualisierung* wird hier eine abstrakte Sichtwei-

---

375) Dieses Kapitel basiert auf Erkenntnissen aus den Beiträgen Dittmann, Schütte et al. (2003), der wiederum maßgeblich auf der Arbeit Schütte, Zelewski (2002) aufbaut, und Dittmann, Penzel (2004). Siehe ebenda zum tieferen Verständnis.

376) Vgl. Albert (1987), S. 45.

377) Vgl. Albert (1987); Albert (1991).

378) Vgl. Vollmer (1994).

379) Vgl. Lorenzen (1987).

380 Vgl. Hartmann, Janich (1996).

381) Vgl. für das Feld der Wirtschaftsinformatik Schütte (1999).

382) Vgl. Thacker, Sheth et al. (2003), S. 281; Gómez-Pérez, Benjamins (1999), S. 1 f.; Studer, Benjamins et al. (1998), S. 184.

383) Vgl. Wand, Monarchi et al. (1995), S. 287; Wand (1996), S. 283 f.

384) So geschehen bspw. bei Chandrasekaran, Josephson et al. (1999), S. 21 und Gómez-Pérez, Benjamins (1999), S. 1.5.

385) Ähnlich verhält es sich auch mit den Modellierungsprimitiven einer Wissensrepräsentationssprache, die bereits von Dritten konzeptualisiert wurde. Siehe hierzu auch Kapitel 7.2, S. 221 ff.

se auf Phänomene eines Realitätsausschnitts verstanden, der für die Erkenntniszwecke der erkennenden Subjekte von Interesse ist. Aus diesen Erkenntniszwecken leiten sich die für die erkennenden Subjekte relevanten Aspekte der wahrgenommenen Phänomene ab. Empirische Untersuchungen zur Informationsmodellierung belegen, welch gravierenden Einfluss Deutungsmuster auf die Konzeptualisierung ausüben.[386] Die persönlichen Erfahrungen, das weitere (erlernte) Wissen und die Interessen eines Erkenntnissubjekts führen dazu, dass es perzeptive oder kognitive Strukturen *erschafft*, die den Ausgangspunkt der Konzeptualisierung darstellen. Konzeptualisierung bedeutet daher immer sowohl die zweck- als auch die subjektabhängige Auszeichnung relevanter Realitätsaspekte. Das Ergebnis eines Konzeptualisierungsprozesses stellen oftmals die *Konzepte* und *Relationen* dar, mit denen der betrachtete Realitätsausschnitt hinsichtlich seiner für relevant erachteten Aspekte *vorstrukturiert* wird. Konzeptualisierung geht also immer mit einer erkenntnisprägenden Vorstrukturierung möglicher Realitätserfahrung einher. Weil die Konzepte als Resultate im Allgemeinen als (natürlich) sprachliche Konstrukte ausgedrückt werden, lässt sich eine Konzeptualisierung auch als eine *begriffliche* Vorstrukturierung möglicher Realitätserfahrung auffassen. Daher wird ein Vokabular, das Repräsentationsbegriffe zur Beschreibung realer Phänomene bereitstellt, oftmals als zentraler Bestandteil von Ontologien angesehen.[387]

Das Ergebnis eines Konzeptualisierungsprozesses stellt dem Verständnis einiger Autoren zufolge kein formalisiertes Modell dar.[388] In dieser Hinsicht folgen sie nicht der „formalistischen“ Konzeptualisierungsdefinition von Genesereth, Nilsson (1987), S. 9 ff., auf die sich auch das Konzeptualisierungsverständnis von Gruber bezieht. Würde eine Konzeptualisierung bereits ein formalsprachliches Artefakt darstellen, dann wäre eine Ontologie, die wiederum als ein formalsprachliches Artefakt aufgefasst wird, nur eine Verdopplung der Konzeptualisierung.

Wird von all den Problemen abgesehen, die während der strukturschaffenden Konzeptualisierung von Realitätsausschnitten zu bewältigen sind, offenbart sich, dass die Anhänger der formalistischen Konzeptualisierungsauffassung zu einer naiv-realistischen Grundhaltung neigen. Deshalb wird in dieser Arbeit als Ergebnis des Prozesses *Konzeptualisierung* nicht notwendigerweise ein formalisiertes Modell angesehen. Es wird vielmehr ein natürlichsprachliches konzeptuelles Modell als Ergebnis angesehen.[389]

Abschließend bleibt anzumerken, dass der Entwicklungsprozess von Ontologien selbst eine zentrale Rolle bei der Entwicklung von Ontologien spielt, weil unabhängig von der zuvor geschilderten Problematik der Konzeptualisierung der Aufbau des Prozesses unmittelbar auf die

386) Vgl. Shanks (1997), S. 65 ff.

387) Vgl. Gruber (1993), S. 199; Staab (2002), S. 201.

388) Vgl. Wand (1996), S. 283; Fernández López (1999), S. 4.8. Einige Autoren benennen zwar nicht explizit den Konzeptualisierungsprozess, sondern sprechen von einer semi-formalen Beschreibung: Angele, Fensel et al. (1998), S. 180 f.; Staab (2002), S. 203 f.

389) Im vorliegenden Kontext der Konzeptualisierung wird der Begriff der natürlichen Sprache weit gefasst und auch auf graphische Ausdrucksmittel übertragen.

Inhalte der Ontologie wirkt. Um den Entwicklungsprozess für Ontologien transparenter und „systematischer" zu gestalten, werden deshalb Vorgehensmodelle eingesetzt.[390]

4.1.4.2 Einsatzgebiete für Ontologien

Es lassen sich mittlerweile zahlreiche Einsatzgebiete für Ontologien ermitteln. So weist Zelewski auf einige typische Anwendungsszenarien in den Wirtschaftswissenschaften hin, wie die Integrationsproblematik der klassischen Funktionalorganisation, inner- und überbetriebliche Integration der Informationsverarbeitungssysteme, computergestützte Gruppenarbeit, elektronische Marktplätze und Multi-Agenten-Systeme.[391]

In dieser Arbeit wird jedoch eine Aufteilung präferiert, die anhand der Sprachen zur Repräsentation von Wissen mittels Ontologien, die sich in „traditionelle" und Markup-Sprachen (auch: Annotationssprachen) unterscheiden lassen, in Einsatzgebiete für Ontologien unterteilt.[392] Während erstere bei herkömmlichen Wissensbasierten Systemen zum Einsatz kommen, lässt sich das Einsatzgebiet für letztere mit dem Schlagwort „Semantic Web" zusammenfassen.[393]

Beispiele für traditionelle Einsatzgebiete von Ontologien lassen sich finden in den Bereichen:[394]

- Wissensmanagement [Lau, Sure (2002); Sure, Maedche et al. (2000); Vasconcelos, Kimble et al. (2000)]
- Electronic Commerce [Labrou (2002), Tsou, Kao (2003)]

---

390) Gemäß Schütte erfordert jedes „wissenschaftlich begründete Vorgehen eine Methode, d. h. ein mittel- und zweckgeleitetes planmäßiges Vorgehen" (Schütte (1998), S. 117). Zu den Vorgehensmodellen zur Ontologiekonstruktion siehe Kapitel 5.1.2.3, S. 136 ff., und zum Ergebnis *OntoFMEA-Vorgehensmodell* zur Entwicklung einer FMEA-Ontologie dieser Arbeit siehe Kapitel 6, S. 193 ff.

391) Vgl. Zelewski (2002a), S. 65.

392) Diese Unterscheidung von Sprachen zur Darstellung einer Ontologie findet sich auch bei Gómez-Pérez, Fernández-López et al. (2004), S. 199 ff. Der Grund für diese Bevorzugung liegt in der Anschlussfähigkeit dieser Unterscheidung zu Kapitel 4.2.1.2.3, S. 104 ff. Siehe zur Unterscheidung auch die Fn. 461, ebenfalls S. 104.

393) Wie bereits in der vorherigen Fn. angemerkt, wurde insbesondere auf die Anschlussfähigkeit des Systematisierungsansatzes geachtet. Im Vordergrund der Ausführungen steht die Verdeutlichung der Anwendungsbreite und -tiefe von Ontologien. Dabei ist es möglich, dass Bereiche (wie bspw. das Wissensmanagement) in beiden Einsatzgebieten angetroffen werden. Für Einsatzgebiete, die jeweils nur einem Bereich zugeordnet wurden, heißt das nicht, dass diese Einsatzgebiete nicht auch für das andere Einsatzgebiet denkmöglich wären. Es bedeutet lediglich, dass der Verfasser hierüber keine Kenntnis erlangen konnte. Dies führt auch zu der Erkenntnis, dass die vorliegende Einteilung lediglich Beispiele zur Verdeutlichung enthält und keinerlei Anspruch auf Vollständigkeit erhebt. Absicht des Verfassers ist hier besonders, den aktuellen Stellenwert von Ontologien zu verdeutlichen.

394) Eine ähnliche Auflistung findet sich in Erdmann (2001), S. 71 f. Allerdings wird dort nicht zwischen traditionellen und Markup-Sprachen unterschieden.

- Ingenieurwesen [Borst (1997); Fernández López, Gómez-Pérez et al. (1999); Kitamura, Sano et al. (2002); Mizoguchi, Kozaki et al. (2000); Pocsai (2000); Schlenoff, Ivester et al. (1998); Uitermark (2001)]

- Sprachverarbeitung/Maschinelle Übersetzung [Knight, Luk (1994); Lenat (1995); Miller, Beckwith et al. (1990); Vossen (2003)]

- Knowledge Engineering[395] [Friedland, Allen (2003); Smith, Becker (1997)]

- Geschäftsprozesse [Kim, Fox (1999); Uschold, Grüninger (1996); Uschold, King et al. (1998)]

Beispiele für Einsatzgebiete von Ontologien im Semantic Web lassen sich finden in den Bereichen:

- Informationssuche [Fensel, Angele et al. (1999); Labrou, Finin (1999); Stuckenschmidt (2003); Theobald (2003); Vögele, Hübner et al. (2003)]

- Wissensmanagement [Motta, Shum et al. (2000); Benjamins, Fensel et al. (1998)]

- Geschäftsprozesse [Staab, Schnurr (1999); van Hoof, Fillies (2003)]

- Multi-Agenten-Systeme [Bailin, Truszkowski (2002); Möller, Haarslev (2003); Sedbrook (2001)]

Diese Einsatzgebiete könnten ohne den Einsatz von IT nicht erfolgreich besetzt werden. Definitorisch verlangt der Einsatz von Ontologien den Einsatz von IT.[396] Um die Vorstellung des Instruments Ontologien abzurunden, wird deshalb im folgenden Kapitel kurz auf die IT-Unterstützung zur Ontologieentwicklung und -verwendung eingegangen.

4.1.4.3 IT-Unterstützung

Zur Entwicklung und Verwendung von Ontologien werden IT-gestützte Werkzeuge verwendet.[397] Zur Entwicklung von Ontologien werden Editoren genutzt. Zur Verwendung von Ontologien werden in der Hauptsache Inferenzmaschinen benötigt.

395) Hierzu gehören bspw. Problemlösemethoden.

396) Siehe hierzu Kapitel 2.1.2.4, S. 28 ff.

397) Übersichten zu IT-gestützten Werkzeugen zur Ontologieentwicklung und -verwendung finden sich bspw. in Corcho, Fernández-López et al. (2003), S. 47 ff.; Fensel (2004), S. 47. ff.; Gómez-Pérez, Fernández-López et al. (2004), S. 296 ff., und Staab, Studer (2004), S. 275 ff. Bei der hier vorgestellten Auswahl wird bereits auf Werkzeuge eingeschränkt, die in der Lage sind, F-Logic-Kodierung zu verarbeiten. Die Begründung für diese Auswahl liegt im Vorgriff auf die Erkenntnisse aus Kapitel 4.2.3, S. 110, und in der dortigen Entscheidung zur Verwendung von F-Logic in dieser Arbeit.

4.1.4.3.1 Entwicklung von Ontologien

- OntoEdit
  OntoEdit[398] stellt eine Ontologieentwicklungsumgebung dar, die als Kern der On-To-Knowledge-Werkzeugsammlung der Ontoprise GmbH das Erstellen, Bearbeiten, Formalisieren und Visualisieren von Ontologien unterstützt. Mit Hilfe von OntoEdit wird die Spezifikation von Konzepten, die unmittelbar zu einer Hierarchie durch is_a-Relationen zwischen den Konzepten angeordnet werden, weiteren Relationen und Regeln sowie Instanzen ermöglicht. Intern werden die Konstrukte in der Werkzeug-spezifischen Repräsentationssprache OXML (in der aktuellen Version 2.0) formalisiert. Es werden jedoch auch die Repräsentationssprachen XML, F-Logic, RDF(S) und DAML+OIL zur Ein- und zur Ausgabe von Ontologien unterstützt.[399]

- Protégé-2000
  Protégé-2000[400] ist eine Open-Source-Software, die als Einzelanwendung installiert wird. Ähnlich wie OntoEdit lässt sich Protégé-2000 mittels Plug-ins erweitern. Ontologien lassen sich in den Formaten F-Logic, OIL, XML, OWL und Prolog ein- und ausgeben.

- WebOde
  Bei WebOde[401] handelt es sich um eine Werkzeugsammlung zur Entwicklung und Verwendung von Ontologien. Der Ontologie-Editor in WebOde basiert auf HTML und erlaubt die Entwicklung von Ontologien und das Browsen durch diese Ontologien. Als Ein- und Ausgabeformate für Repräsentationssprachen werden unterstützt: XML, RDF(S), OIL, DAML+OIL, OWL, F-Logic und Prolog. WebOde kann mit Protégé-2000 in Kooperation arbeiten.[402]

Grundsätzlich lässt sich für die Kodierung einer Ontologie auf die genannten Hilfsmittel verzichten. Die Syntax von F-Logic bspw. erlaubt aufgrund ihrer Benutzerfreundlichkeit die „direkte“ Programmierung mit einem einfachen Textverarbeitungsprogramm.[403]

4.1.4.3.2 Verwendung von Ontologien

Die Abbildung 21 verdeutlicht als Skizze das Grundkonzept eines Wissensbasierten Systems auf der Basis von Ontologien und FMEA-Wissen.[404] Grundsätzlich besteht das System aus

398) Aktuelle Informationen zu OntoEdit und zur On-To-Knowledge-Werkzeugsammlung finden sich unter http://www.ontoprise.de, Zugriff am 17.03.2007.

399) Zu den Sprachen zur Repräsentation von Ontologien siehe Kapitel 4.2, S. 95 ff.

400) Vgl. Noy, Fergerson et al. (2000). Zu aktuellen Informationen siehe http://protege.stanford.edu, Zugriff am 18.03.2007.

401) Vgl. Arpírez, Corcho et al. (2001).

402) Vgl. Gómez-Pérez, Fernández-López et al. (2004), S. 328 und Corcho, Gómez-Pérez (2004).

403) Zu einem Überblick über die Syntax von F-Logic siehe Kapitel 7.2.3, S. 223 ff.

den Bestandteilen *Abfragesystem*, *Wissensbasis*, *FMEA-Ontologie* und möglichen *Quellen* für die Wissensakquisition. Das Abfragesystem wird wiederum aus der *Benutzerschnittstelle*, dem *Webserver* und der *Inferenzkomponente* zusammengesetzt.

Über die Benutzerschnittstelle interagiert der Benutzer mit dem Wissensbasierten System auf der Grundlage von Ontologien und FMEA-Wissen. Die FMEA-Ontologie, oder auch nur Teile von ihr, können genutzt werden, um dem Benutzer die Arbeit mit dem System zu erleichtern. Bspw. wäre es möglich, dem Benutzer die Konzepte, die hierarchisch taxonomisch miteinander verbunden sind, als Baumstruktur zum Aufklappen und Auswählen zur Verfügung zu stellen. Diese Wissensbereitstellung erhöht den Komfort für den Benutzer merklich im Gegensatz zur Abfrage des Systems mittels einzeln einzutippender Abfragen (Queries).

Der Webserver dient als Bindeglied zwischen der Benutzerschnittstelle und der Inferenzkomponente. Eine Anfrage über die Benutzerschnittstelle wird mittels des eingesetzten Webservers an die Inferenzkomponente weitergeleitet, die eine Antwort aus der Wissensbasis unter Einsatz der Ontologie ableitet.[405] Der Webserver ermöglicht die verteilte Kommunikation mit dem System über das Internet. In der Wissensbasis befindet sich das Wissen der bereits durchgeführten und eingegebenen FMEA-Formblätter (Instanzenwissen).[406]

Die (FMEA-)Ontologie, in dieser Arbeit repräsentiert in F-Logic, bildet zusammen mit der Wissensbasis, die ebenfalls in F-Logic kodiert wird, den Kern des Systems.[407]

Die Inferenzkomponente greift über das F-Logic-Programm sowohl auf die Wissensbasis als auch auf die Ontologie zu.

404) Vgl. hierzu auch Abbildung 4, Seite 25. Deutlich wird, dass die Ontologie an der Wissensbasis ansetzt und nicht zur Wissensbasis gezählt wird. Die Abbildung 21 skizziert lediglich ein laufendes System, d. h. die Entwicklerschnittstelle wurde außer Acht gelassen. Die Dialogkomponente findet sich im Webserver wieder. Als besonderer Vorteil von Ontologien gilt deren Mächtigkeit bei der Erklärung von Problemlösungswegen durch Nennung der verwendeten Regeln beim Schlussfolgern. Im skizzierten System wird die Erklärungskomponente der Inferenzkomponente zugerechnet. Die Wissensakquisition wird sowohl mittels „harter“ Eingabe von Hand als auch über die Anwendungs-/ Datenschnittstellen sichergestellt.

405) Siehe zu dieser Funktionsweise den Protoyp OntoFMEA in Kapitel 8, S. 252 ff.

406) Siehe hierzu Kapitel 2.1.2.4, S. 28 ff, und hier insbesondere die Fn. 122, S. 28.

407) Erdmann bezeichnet das Artefakt aus Ontologie und Wissensbasis in F-Logic als F-Logic-Programm (vgl. Erdmann (2001), S. 98). Diesem Begriffssystem folgt der Verfasser.

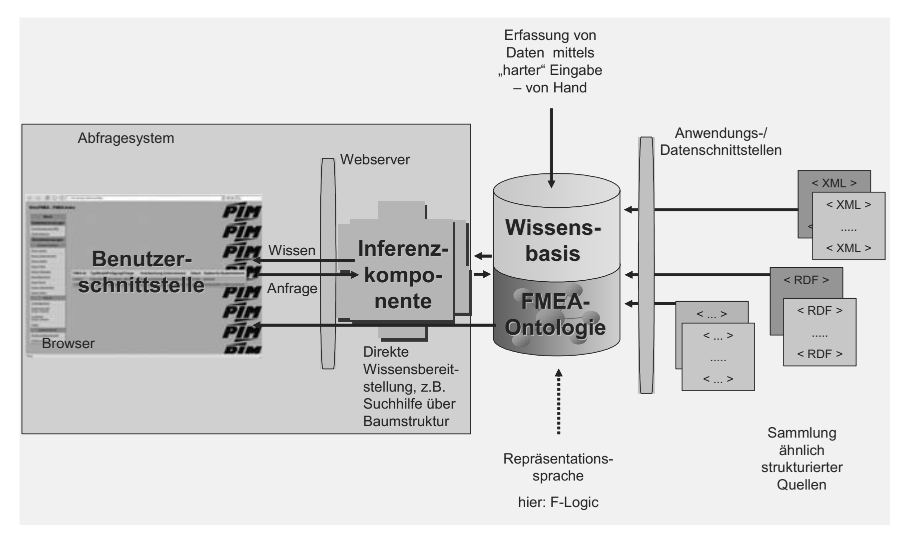


Abbildung 21: Wissensbasiertes System auf Basis von Ontologien und FMEA[408]

Die in dieser Arbeit verwendete Inferenzkomponente Ontobroker[409] kann Antworten auf Anfragen in F-Logic berechnen. F-Logic erweitert dabei Horn-Programme um Modellierungsprimitive von Frames (siehe Kapitel 4.1.3.2.5, S. 84). Die Inferenzmaschine Ontobroker basiert auf einer Beweistheorie, die die „Well-Founded Model Semantic“[410] von generalisierten Horn-Programmen errechnet.[411] Zu diesem Zweck werden die Konstrukte des F-Logic-Programms von der Inferenzmaschine in ein generalisiertes Horn-Programm übersetzt.[412] Hierzu greift die Inferenzmaschine auf Erkenntnisse von Decker und Fensel zurück, die durch

408) In Anlehnung an die OntoBroker-Architektur nach Studer, Abecker et al. (1999), S. 271.

409) Zu aktuellen Informationen siehe http://www.ontoprise.de, Zugriff am 20.01.2007. Für die Verarbeitung von F-Logic-Programmen finden sich derzeit lediglich zwei weitere Inferenzmaschinen: Florid (F-Logic Reasoning in Databases; vgl. Ludäscher, Himmeröder et al. (2000), S. 3 ff.; May (2000), entwickelt am Institut für Informatik der Universität Freiburg) und Flora(2) (vgl. http://flora.sourceforge.net/, Zugriff am 25.02.2005), das als Anwendung eine Einbindung der Inferenzmaschine XSB (vgl. http://sourceforge.net/projects/xsb/, Zugriff am 18.03.2007) erfordert, entwickelt am Institut für Informatik der Universität New York/Stony Brook. Der Hauptgrund für die Verwendung von Ontobroker in dieser Arbeit lag darin, dass am Institut für Produktion und Industrielles Informationsmanagement an der Universität Duisburg-Essen (Campus Essen) mit der Inferenzmaschine Ontobroker bereits gearbeitet wurde. Grundsätzlich geht der Verfasser jedoch von einer Übertragbarkeit seiner Ergebnisse auf die beiden anderen Inferenzmaschinen aus, weil die verwendeten Modellierungsprimitive in allen von den Inferenzmaschinen verarbeitbaren Dialekten vorkommen. Auch geht der Verfasser davon aus, dass seine Ontologie mit relativ geringem Aufwand in weiteren regelbasierten Systemen Verwendung finden kann.

410) Vgl. van Gelder, Ross et al. (1991).

411) Vgl Erdmann (2001), S. 98. Zu Horn-Programmen und deren Semantik vgl. Schöning (2000), S.120 ff.

412) Vgl. Stuckenschmidt, van Harmelen (2005), S. 188.

geeignete Transformationsschritte zeigen konnten, wie F-Logic-Programme in generalisierte Horn-Programme transformiert werden können.[413]

Am Institut für Produktion und Industrielles Informationsmanagement der Universität Duisburg-Essen (Campus Essen) wurde vom Verfasser auf der Grundlage der voranstehenden Ausführungen ein Prototyp eines webbasierten Softwaresystems mit dem Namen OntoFMEA entwickelt, das in Kapitel 8, S. 252 ff., vorgestellt wird.[414]

Grundvoraussetzung für die Verwendung einer Wissensbasis und von Ontologien ist deren Darstellung in einer Wissensrepräsentationssprache, auf die im anschließenden Kapitel eingegangen wird.

413) Vgl. Erdmann (2001), S. 98. Zu den Erkenntnissen von Decker, Fensel et al. vgl. Decker (1998), S. 9.3 ff., und Fensel, Decker et al. (1998), S. 19 ff. Weitere Informationen zur Arbeitsweise von Ontobroker finden sich in Angele, Lausen (2004), S. 45 f., und Fensel (2004), S. 82.

414) Vgl. hierzu auch die Ausführungen in Dittmann, Rademacher (2004); Dittmann, Rademacher et al. (2004) und Dittmann, Zelewski (2004).

4.2 Repräsentation von Wissen in Ontologien

4.2.1 Gliederung von Repräsentationssprachen

4.2.1.1 Gliederung von Repräsentationssprachen in der Literatur

Das Kapitel 4.2 dient der Gliederung von Repräsentationssprachen zur Darstellung von Ontologien und der Festlegung auf eine Repräsentationssprache, die in der vorliegenden Arbeit einer weiteren Verwendung, d. h. der Darstellung der FMEA-Ontologie, zugeführt wird.[415]

Die Repräsentation von Wissen gilt als eines der Schlüsselprobleme bei der Entwicklung Wissensbasierter Systeme im Speziellen und der Forschung zur Künstlichen Intelligenz im Allgemeinen.[416] Insbesondere wird hierbei auf die Möglichkeiten der formalen Logik zurückgegriffen. Die formale Logik hat für die Wissensrepräsentation aufgrund ihrer Grundforderung nach Eindeutigkeit, Präzision und ihres hohen Grades an Effizienz bei Schlussfolgerungen in formalisierbaren Bereichen grundlegende Bedeutung. Bibel etwa hält den Logikformalismus für kanonisch für das Fachgebiet der Wissensrepräsentation innerhalb der Forschungen zur Künstlichen Intelligenz und der Kognitionswissenschaft.[417] Auf Seite 32 derselben Quelle geht Bibel sogar so weit, die „ ... Sprache der Mathematik als Formalismus zur Repräsentation von Wissen unter die Logik subsumieren [zu] können und nicht eigens behandeln [zu] müssen“.[418]

Grundsätzlich wird jede Sprache durch zwei Aspekte charakterisiert: ihre Syntax und ihre Semantik. Die Syntax befasst sich zum einen mit dem Aufbau von Grundzeichen oder Grundausdrücken einer Sprache, zum anderen mit der Erzeugung von Sätzen aus diesen Grundzeichen oder Grundausdrücken. Demgegenüber beschäftigt sich die Semantik einer Sprache mit der Bedeutung der Grundzeichen oder Grundausdrücke und den Bedingungen, unter denen Sätze dieser Sprache als wahr angenommen werden.[419]

Für einen Überblick werden zunächst Repräsentationssprachen mit Hilfe einiger Gliederungen aus wissenschaftlichen Arbeiten in Kategorien unterschieden.

Puppe unterscheidet ursprünglich Repräsentationsansätze in Regeln, Frames, Constraints, probabilistisches Schließen, nicht-monotones Schließen und temporales Schließen.[420] Zusätz-

415) Darüberhinaus wird diese FMEA-Ontologie in der verwendeten Sprache im Prototyp OntoFMEA eingesetzt.

416) Vgl. Vámos (1998), S. 3.1. Gleichwohl steckt die Wissensrepräsentation noch in den Anfängen, so dass Gliederungen des Gebiets immer Herausforderungen an die Autoren stellen, weil man sich nicht an kanonischen Gliederungsmustern zu orientieren vermag (vgl. Bibel (1993), S. 16).

417) Vgl. Bibel (1993), S. 19. Zusammengefasst werden die Bereiche Künstliche Intelligenz und Kognitionswissenschaft von Bibel als Intellektik bezeichnet.

418) Bibel berücksichtigt dabei jedoch nicht die hierzu im Gegensatz stehenden Erkenntnisse der Kritik an B. Russell und seines Ansatzes die Mathematik vollständig auf Basis der Logik zu definieren (bspw. Russell´s Antinomie: beinhaltet die Menge aller Mengen sich selbst?).

419) Vgl. Beckermann (1997) S. 51.

420) Vgl. Puppe (1991), S. 16 ff.

lich berücksichtigt er die Prädikatenlogik als „häufig verwendeten Bezugspunkt“.[421] Als Grundtechniken identifiziert er Regeln und Frames.[422] Weil lediglich eine Übersicht dargestellt werden soll, wird im Folgenden nur auf den „Bezugspunkt“ und die „Grundtechniken“ rekurriert:[423]

- Bezugspunkt *Logik*
  Ein Kalkül[424] der (Prädikaten-)Logik soll Objekte und deren Zustände in der realen Welt durch Aussagen des Kalküls beschreiben und mit allgemeingültigen Ableitungsregeln andere Aussagen herleiten. Diese hergeleiteten Aussagen werden wiederum auf Objekte der realen Welt bezogen.[425]

- Grundtechnik *Regeln*
  Ein regelbasiertes System gemäß Puppe enthält hauptsächlich eine Wissensbasis, die aus gültigen Fakten und Regeln zur Herleitung neuer Fakten besteht, und einen Regelinterpreter, der den Herleitungsprozess steuert.[426] Regeln bestehen aus einer „Vorbedingung“ und einer „Aktion“. Ist die Vorbedingung erfüllt, so wird die Aktion ausgeführt. Grundsätzlich lassen sich zwei Aktionsweisen unterscheiden. Zum einen wird durch „Induktionen“ und „Deduktionen“ der Wahrheitsgehalt einer Feststellung ermittelt und zum anderen werden mit „Handlungen“ Aussagen über Veränderungen von Zuständen der realen Welt getroffen.[427] Das zu erhaltende (zu explizierende) Wissen stellen die Regeln (in Verbindung mit den Fakten) dar, d. h. implizit ist bereits das Wissen innerhalb der Regeln und Fakten repräsentiert. Die Datenbasis ist oftmals eine unstrukturierte Menge von Fakten.

- Grundtechnik *Objekte/Frames.* Aus Gründen der Vereinfachung und Nachvollziehbarkeit spielen, im Gegensatz zum regelbasierten Ansatz, die Formalismen zur Strukturierung der Fakten die tragende Rolle.[428] Es werden Vererbungshierarchien, zugeordnete Operationen und Default-Werte ermöglicht. Ein „Frame“ ist eine formularähnliche Objektbeschreibung (Datenstruktur) mit Merkmalen (synonym: Objekteigenschaften, „slots“) und Merkmalsausprägungen (synonym: Werte, „values“).[429]

421) Vgl. Puppe (1991), S. 16.

422) Vgl. Puppe (1991), S. 11.

423) Der Bezugspunkt *Logik* dient dabei als Basis für die beiden Grundtechniken, d. h. mittels prädikatenlogischer Formulierungen werden Regeln und Fakten ausgedrückt.

424) Im Allgemeinen versteht man unter einem Kalkül ein formales Regelsystem. Die Anwendung der Regeln kann dabei rein syntaktisch erfolgen.

425) Vgl. Pocsai (2000), S. 23.

426) Vgl. Puppe (1991), S. 22.

427) Vgl. Puppe (1991), S. 21.

428) Inwiefern diese Formalismen eine Vereinfachung und Nachvollziehbarkeit unterstützen, wurde bereits im Kapitel 4.1.3.2.5, S. 84 f., kurz dargestellt.

429) Vgl. Pocsai (2000), S. 24.

Ostermayer unterscheidet neben einer Unterteilung von Wissen in deklarativ und prozedural vier Ansätze der Wissensrepräsentation:[430]

- *Assoziative Netze* mit den Vertretern Semantische Netze, Begriffliche Graphen (Conceptual Graphs) und Begriffliche Abhängigkeiten.
- *Strukturierte Objekte* mit den Vertretern Frames, Schemata, Units und Objekte.
- *Logikbasierte Repräsentationen* mit den Formen Aussagenlogik, Prädikatenlogik, Modallogik, Mehrwertige (Multivalued) Logik und Unscharfe (Fuzzy) Logik sowie sonstige Unterarten von Logiken.
- *Prozedurale Repräsentationen* wie zum Beispiel Produktionsregeln, Constraints und Scripts.

Haun nennt in seiner Klassifikation in einer Art Baumstruktur in der obersten Ebene deklarative und prozedurale (sowie eingeschränkt die Hybris aus beiden) Ansätze. Die deklarativen Ansätze unterteilt er wiederum in traditionelle datenmodellbasierte Ansätze, objektorientierte Wissensrepräsentationsansätze, regelbasierte Ansätze und lexikalische Ansätze[431]. Zur prozeduralen Wissensrepräsentation zählt er insbesondere Agenten und Programme.[432]

Die genannten Ordnungsmöglichkeiten von Puppe, Ostermayer und Haun beruhen auf syntaktischen Ordnungskriterien. Den Ansätzen liegen zumeist (eingeschränkte) Logikkalküle zugrunde. So unterscheidet Puppe bspw. explizit zwischen *Logik* als Bezugspunkt und *Regeln* und *Frames* als Grundtechniken, die auf Logik als gemeinsamem Bezugspunkt basieren. Bspw. werden Produktionsregeln für den Einsatz zur Steuerung von Produktionssystemen auf der Basis von eingeschränkten Logikkalkülen computerverarbeitbar dargestellt.[433] Logik im Sinne einer formalen Semantik wird gebraucht, damit nicht nur Informationen gefunden, gefiltert und aufgearbeitet werden können, sondern Wissen aus einem Bestand durch Schlussfolgerungen expliziert werden kann. Es wird deutlich, dass es einer zugrunde liegenden Logik bei der Repräsentation von Wissen bedarf.

---

430) Vgl. Ostermayer (2001), S. 46 ff.

431) Zum Begriff „lexikalisch“ siehe auch Fn. 878, S. 247.

432) Vgl. Haun (2000) S. 42.

433) Vgl. Beierle, Kern-Isberner (2003), S. 71. Als weiteres Beispiel zur Verdeutlichung sei die Datenbank-Modellierung genannt. Hier weist Bibel jedoch darauf hin, dass den meisten Datenbank-Entwicklern 1993 das Wort „Logik“ offenbar noch unbekannt ist, obwohl sie implizit Logik bei der Entwicklung von Datenbanken verwenden (vgl. Bibel (1993), S. 96).

4.2.1.2 Gliederung von Repräsentationssprachen in dieser Arbeit

4.2.1.2.1 Anforderungen an eine Repräsentationssprache in dieser Arbeit

Wie bereits in Kapitel 4.1, S. 76, dargelegt wurde, wird insbesondere die Möglichkeit des Automatischen Schließens als konstituierendes Merkmal von Ontologien angesehen.[434] Logik erlaubt die Axiomatisierung[435] und das Aufstellen von expliziten Regeln (Formeln zur Schlussfolgerung) von Wissen einer Domäne und das Schlussfolgern auf diesem Wissen. Das Schlussfolgern wird mittels Logikkalkülen, die computerverarbeitbaren Formen der Darstellung von Wissen (Kode) zugrunde liegen, ermöglicht. Die Leistungsfähigkeit einer Inferenzmaschine hängt dabei von der zugrunde liegenden Logik und dem implementierten Schlussfolgerungsalgorithmus ab.[436] Logische Kalküle werden, weil sie als Grundlage dienen, für die weitere Gliederung von Repräsentationssprachen für Ontologien in dieser Arbeit herangezogen. Zusätzlich wird die Gliederung aus Kapitel 4.1.4.2, S. 89 ff., aufgegriffen, die traditionelle Spezifikationssprachen und spezielle Spezifikationssprachen (Markup-Sprachen für das Semantic Web) unterscheidet. In den folgenden Unterkapiteln werden bekannte Spezifikationssprachen gemäß dieser Zweiteilung und der Gliederung logischer Kalküle kurz beschrieben und eingeordnet. Um anschließend anhand der wissenschaftlichen Problemstellung dieser Arbeit eine Spezifikationssprache auszuwählen, die die Grundlage für die Problemlösung dieser Arbeit darstellt, müssen einige zentrale Anforderungen an diese Spezifikationssprache gestellt werden.

Die zentralen Anforderungen in dieser Arbeit an eine Sprache zur Wissensrepräsentation, die als apriorisch in Bezug auf das hier vorliegende Ontologieverständnis anzusehen sind, lassen sich zusammenfassen zu:[437]

- *Computerverarbeitbarkeit*
  Aus der Problemstellung ergibt sich die Notwendigkeit, Ontologien zu erstellen und zu verarbeiten; das bedeutet: eine Sprache zur Darstellung von Ontologien muss für die Konzeption des Wissensbasierten Systems die Implementierung von Ontologien und

434) In diesem Sinne folgt der Verfasser der Auffassung Newells, dass – frei übersetzt – „eine nicht-logische Analyse von Wissen Blödsinn ist“ (Newell (1981), S. 17). Sowa folgt ebenfalls dieser Überzeugung, indem er die Wissensrepräsentation als multidisziplinäres Vorhaben beschreibt, das Theorien und Techniken der Felder Logik, Ontologie (im philosophischen Sinne) und Computertechnik (Computation) miteinander verbindet (vgl. Sowa (2000), S. XI f. und S. 408 ff.).

435) Axiomatisierung innerhalb eines positiven Logikkalküls bedeutet die Aufstellung von allgemeingültigen Formeln (Tautologien), d. h. diese Formeln sind nicht falsifizierbar (vgl. Beierle, Kern-Isberner (2003), S. 38).

436) Vgl. bspw. für die Programmiersprache Prolog Schöning (2000), S. 143 ff.; für einen Vergleich unterschiedlicher ontologiebasierter Systeme mit Inferenzmaschinen vgl. Friedland, Allen (2003). Vgl. ferner Blau (1978), der mit seinem Werk die Vorteilhaftigkeit einer formalen dreiwertigen Logik (wahr/falsch/unbestimmt) gegenüber der weiter verbreiteten zweiwertigen Logik (wahr/falsch) zur formalen Repräsentation der natürlichen Sprachen aufzeigen will.

437) Die Kriterien werden nach der an dieses Kapitel anschließenden Vorstellung von Sprachen zur Repräsentation von Wissen gebraucht, um eine Sprache für die weitere Verwendung in dieser Arbeit auszuwählen. In dem Kapitel 4.2.3 zur Auswahl, S. 110 f., findet sich eine weitergehende Operationalisierung der hier vorgestellten apriorischen zentralen Anforderungen an eine Repräsentationssprache.

damit ihre formale Verarbeitbarkeit ermöglichen. Die zu verwendende Sprache muss formal verarbeitbar sein, um „neues“ Wissen formal mit Hilfe eines Computers explizieren zu können.

- *Ausdrucksmächtigkeit hinsichtlich der Darstellung einer Ontologie*
  Bei der Darstellung von Ontologien wird zwischen leichtgewichtigen und schwergewichtigen Ontologien unterschieden. Verkürzt wird zum einen die *Ausdrucksmächtigkeit einer Spezifikationssprache zur Darstellung von Ontologien ohne Regeln* (leichtgewichtige Ontologien) als Kriterium unterschieden. Auf einer Skala mit den Ausprägungswerten *niedrig*, *mittel*, *hoch* und *sehr hoch* werden die einzelnen Sprachen bewertet. Zum anderen wird als Kriterium angewendet, ob eine Spezifikationssprache *Regeln* (und damit schwergewichtige Ontologien) darstellen kann oder nicht.

- *Nachvollziehbarkeit*
  Das Kriterium der Nachvollziehbarkeit verlangt, dass das repräsentierte Wissen innerhalb einer Sprache *intersubjektiv nachprüfbar* sein muss.[438] Auch Dritten gegenüber soll das Wissen einer Ontologie kommunizierbar und kontrollierbar sein. Auf einer Skala mit den Ausprägungswerten *unübersichtlich*, *mittel*, *hoch* und *sehr hoch* werden die einzelnen Sprachen bewertet.

Ferner wird berücksichtigt, ob eine verteilte Anwendbarkeit (im Sinne der Markup-Sprachen des Semantic Webs) ermöglicht wird und ob es sich um eine originäre Entwicklung für den Bereich der Ontologien handelt, d. h., es wird untersucht, ob es sich um eine Wissensrepräsentationssprache, die schon auf anderen Gebieten eingesetzt wurde, handelt oder nicht.[439]

4.2.1.2.2 Gliederung anhand logischer Kalküle

Die folgende Aufzählung, die für die Gliederung von Repräsentationssprachen für den speziellen Verwendungszweck der Darstellung von Ontologien zugrunde gelegt wird, gibt in et-

---

438) Dieses Kriterium ergibt sich unmittelbar aus dem Ziel für den Einsatz von Ontologien, der Wiederverwendung von Wissen, und den damit verbunden Implikationen, wie bspw. divergierende Wissenshintergründe zu kompensieren. Siehe hierzu Kapitel 4.1.1, S. 74 f.

439) Diese beiden Kriterien (*verteilte Anwendbarkeit* und *originäre Ontologieentwicklung*) dienen im Folgenden lediglich dem weiteren Verständnis. Die Kriterien sollen helfen, einen „Gesamteindruck“ für den Bereich der Sprachen zur Repräsentation von Wissen in Ontologien zu gewinnen. Für die Auswahl einer geeigneten Repräsentationssprache spielen die Kriterien lediglich eine untergeordnete Rolle. Zur Auswahl und ihrer Entscheidungsgrundlage siehe Kapitel 4.2.3, S. 110 f.

wa[440] in aufsteigender Reihenfolge die relative Mächtigkeit der Ausdrucksstärke der Kalküle wieder:[441]

- Aussagenlogik (AL),
- Beschreibungslogik (auch: Description Logic (DL)),
- Prädikatenlogik erster Stufe (PL1; auch: First Order Logic (FOL)),
- Rahmenlogik (auch: Frame Logic (FL)) und
- Prädikatenlogik zweiter Stufe oder höherwertige Logiken (PL2; auch: Higher Order Logic (HOL).

Die Syntax der *Aussagenlogik* besteht aus atomaren Formeln, die verkürzt zu A, B, C, ... dargestellt werden.[442] In der Aussagenlogik werden zusammengesetzte Formeln – wie auch in den anderen Logiken – mittels logischer Operatoren dargestellt. Für die Aussagenlogik gebräuchlich sind die Operatoren:[443]

- Negation, mit dem Zeichen: $\neg$
- Konjunktion, mit dem Zeichen: $\wedge$
- Adjunktion, mit dem Zeichen: $\vee$
- Subjunktion, mit dem Zeichen: $\rightarrow$
- Bijunktion, mit dem Zeichen: $\leftrightarrow$

Jeder atomaren Formel wird in der Aussagenlogik als Semantik ein Wahrheitswert zugewiesen, dabei ist eine Formel entweder wahr oder falsch.[444] Vorteilhaft bei der Aussagenlogik ist

440) Zu den relativen Mächtigkeiten und deren Einschränkungen vgl. die folgenden Ausführungen.

441) Vgl. auch Strang (2003), S. 6. Die hier berücksichtigten logischen Kalküle stellen lediglich eine Auswahl vorhandener Kalküle dar. Sie wurden zum Zwecke der Einteilung von Wissensrepräsentationssprachen ausgewählt. Implizit wird dabei eine Unterscheidung von formaler Logik und mathematischer Logik vorgenommen (so ebenfalls – jedoch explizit – Wolff (2004), der in seiner Abhandlung die Gleichsetzung von formaler und mathematischer Logik konsequent trennt und aufzeigt, dass „mathematische und formale Logik als systematisch verschiedene Teile der deduktiven Logik anzusehen [sind]“, S. XIII). So werden hier in Anlehnung an Wolff bspw. syllogistische Ansätze der formalen Logik ausgelassen, weil der Verfasser a priori möglichen Wissensrepräsentationssprachen aufgrund der geforderten Computerverarbeitbarkeit eine größere Nähe zu „mathematischen“ Ansätzen unterstellt.
Die erwähnte aufsteigende Reihenfolge wird für die Erläuterung einer Beschreibungslogik durchbrochen. Die Erläuterung einer Beschreibungslogik wird hinsichtlich der enthaltenen Ausdrucksmächtigkeit aus Gründen der Argumentation (wichtige Modellierungsprimitive aus der Prädikatenlogik sind Basis einer Beschreibungslogik) erst nach der Erläuterung zur Prädikatenlogik der ersten Stufe behandelt.

442) Bspw. wird die Aussage: *„Spindelschrauber ist defekt“* verkürzt zu *A* zusammengefasst.

443) Implikation und Äquivalenz stellen eigentlich nur Abkürzungen zur besseren Handhabbarkeit dar, weil sie sich mittels der vorgenannten Operatoren darstellen lassen: $(A \Rightarrow B)$ entspricht $(\neg A \vee B)$ und $(A \Leftrightarrow B)$ entspricht $((A \wedge B) \vee (\neg A \wedge \neg B))$.

444) Diese Bedingung ist auch bekannt als *Satz vom ausgeschlossenen Dritten* (tertium non datur).

die „einfache" Überprüfbarkeit der zusammengesetzten Formeln auf Gültigkeit. Es existieren automatisierte Verfahren, die in endlicher Zeit eine Lösung finden.[445] Die Aussagekraft ist aufgrund der beschränkten Ausdrucksmächtigkeit jedoch gering. Für die meisten Anwendungsfälle reichen die Möglichkeiten der Aussagenlogik nicht aus. So ist es bspw. nicht möglich, Beziehungen zwischen zwei objektsprachlichen Aussagen oder Existenzaussagen anzugeben. Für mächtigere Logiken werden weitere Operatoren benötigt und verwendet.

In der *Prädikatenlogik* werden vor allem Quantoren als logische Operatoren hinzugenommen:[446]

- Allquantor, mit dem Zeichen: $\forall$
- Existenzquantor, mit dem Zeichen: $\exists$

Die aus der Aussagenlogik bekannten atomaren Formeln werden durch Prädikatensymbole[447] der Form P_i^k ersetzt, *i* gibt dabei den identifizierenden Index (mit i=1,2,3,...) und *k* (mit k=0,1,2,3,…) die Stelligkeit des Prädikats an. Ferner werden Variablen x_i, Funktionssymbole f_i^k und Terme t_k eingeführt. Für Terme gilt:

1. Jede Variable ist ein Term.
2. Falls f ein Funktionssymbol der Stelligkeit *k* ist und falls $t_1,...t_k$ Terme sind, so ist auch $f(t_1,...,t_k)$ ein Term. 0-stellige Funktionssymbole (mit k=0) werden als Konstanten behandelt.

Formeln in der Prädikatenlogik lassen sich induktiv definieren:

1. Falls P_i^k ein Prädikatensymbol der Stelligkeit *k* ist und falls $t_1,...t_k$ Terme sind, dann ist $P(t_1,...,t_k)$ eine Formel.
2. Für alle Formeln *F* und *G* sind (F→G), (F↔G), $(F \wedge G)$ und $(F \vee G)$ Formeln.
3. Für jede Formel *F* ist $\neg F$ eine Formel.
4. Falls *F* eine Formel ist und *x* eine Variable, die in der Formel *F* enthalten ist, dann sind auch $\forall xF$ und $\exists xF$ Formeln.

445) Bspw. lässt sich eine solche Lösung mittels eines sehr effizienten Erfüllbarkeitstests für Hornformeln (vgl. Schöning (1992), S. 32 ff.) finden. Dabei wird eine Formel als Hornformel bezeichnet, wenn sie in konjunktiver Normalform vorliegt und jedes Adjunktionsglied höchstens ein positives Literal enthält.

446) Zusätzlich werden Funktions- und Prädikatssymbole eingeführt. Zu einer Einführung und Übersicht in die Prädikatenlogik vgl. Epstein (1994), S. 50 ff. In weiteren Logiken, die in dieser Arbeit keine Berücksichtigung finden, lassen sich noch mehr Operatoren finden, wie etwa in der Modallogik für Modalitäten.

447) In dieser Arbeit wird ein Symbol als eine Bezeichnung definiert, die aus beliebigen Zeichen besteht und der eine Bedeutung beigemessen wird. Das heißt, ein Symbol wird als Zeichen oder Wort (Zeichenkette), dem eine Bedeutung beigemessen wird, angesehen.

Die Prädikatenlogik gilt als die am genauesten untersuchte und allgemein angewandte Logik (in den Wissenschaften).[448] Zentral für die Zwecke Wissensbasierter Systeme ist das Testen von Formeln auf Erfüllbarkeit oder Allgemeingültigkeit durch entsprechende Algorithmen. Aus prinzipiellen Gründen existieren nicht für alle prädikatenlogischen Formeln derartige Algorithmen. Die Prädikatenlogik gilt per se als unentscheidbar.[449] Es existieren lediglich so genannte Semi-Entscheidungsverfahren.[450] Hinzu kommt, dass der eigentliche Schlussfolgerungsprozess in der Regel ungeeignet für eine Begründung im Sinne einer Erklärung ist, weil kaum zielgerichtete Entscheidungsverfahren existieren. Um diesem gravierenden Mangel bei der computergestützten Verwendung prädikatenlogischer Formeln entgegenzutreten, wurden eingeschränkte Kalküle entwickelt, die eine Entscheidbarkeit in endlicher Zeit ermöglichen.

Unter *Beschreibungslogik(en)* werden mehrere solcher eingeschränkter Logikkalküle zusammengefasst.[451] Jedoch basieren alle auf dem „Ursprungskalkül" ACL und bilden lediglich Erweiterungen hiervon durch Darstellung zusätzlicher Operatoren. Beschreibungslogik erlaubt die Spezifikation einer terminologischen Hierarchie (Taxonomie) von Konzepten mittels einer eingeschränkten Menge von Formeln der Prädikatenlogik erster Stufe. Eine Beschreibungslogik unterscheidet und erlaubt lediglich ein- und zweistellige Prädikate. Die einstelligen Prädikate oder *Konzepte* (atomic concepts) entsprechen herkömmlichen Klassen. Die zweistelligen Prädikate beschreiben die Beziehungen zweier Konzepte zueinander als eine *Rolle* (atomic role), vergleichbar mit Relationen. In ACL werden hierzu noch fünf Operatoren definiert:

- Konzeptkonjunktion, interpretiert als Schnittmenge zweier Konzepte,
- Konzeptdisjunktion, interpretiert als Vereinigungsmenge,
- Konzeptnegation,
- Universelle Quantifikation: $\forall$ R:C und
- Existentielle Quantifikation: $\exists$ R:C.[452]

448) Vgl. Sowa (2000), S. 18.

449) Zur Unentscheidbarkeit der Prädikatenlogik der ersten Stufe vgl. Siefkes (1990), S. 198 ff.

450) Vgl. hinsichtlich der zu großen Ausdrucksmächtigkeit der Prädikatenlogik erster Stufe und der damit einhergehenden Unentscheidbarkeit und Semi-Entscheidungsverfahren Schöning (1992), S. 71 ff., und Owsnicki-Klewe, von Luck et al. (2003), S. 171.

451) ACL gilt auch als ein möglicher „Dialekt" von Beschreibungslogiken. Eine weitere bekannte Beschreibungslogik stellt SHIQ dar, die den webbasierten Repräsentationssprachen OIL, DAML+OIL und OWL als Basis dient (vgl. Baader, Horrocks et al. (2004), S. 3; Horrocks, Patel-Schneider et al. (2003), S. 9 ff.). SHIQ umfasst ACL und erweitert dieses Kalkül um transitive Rollen, inverse Rollen und zahlenmäßige Restriktionen.
Häufig findet sich in der deutschsprachigen Literatur auch der englische Fachterminus *Description Logic(s)* (DL). Zur Einführung in die Thematik siehe Baader, Horrocks et al. (2004); Baader, Calvanese et al. (2003).

452) Universelle und existenzielle Quantifikation erfolgt ähnlich zur Prädikatenlogik, indem ein Rollenfüller R angegeben wird für ein Konzept C, der entweder immer gültig ist für das Konzept oder anzeigt, dass es zumindest einen Rollenfüller gibt, der zu einem bestimmten Konzept gehört (vgl. Owsnicki-Klewe, von Luck et al. (2003), S. 173).

Beschreibungslogiken unterscheiden zwischen einer Tbox und einer Abox. Tbox und Abox ergeben zusammen die Wissensbasis (Knowledge Base). Die Tbox (t für *terminological knowledge*) legt die Terminologie und Struktur der Konzepte zueinander fest, dabei werden Terme hierarchisch zu Konzepten geordnet. Die Abox (a für *assertional knowledge*) stellt Behauptungen auf, indem sie Konzepte und Rollen der Tbox mit Objekten (auch Individuennamen genannt) verbindet.[453]

Beschreibungslogiken werden hinsichtlich ihrer Ausdrucksmächtigkeit zwischen der Aussagenlogik und der Prädikatenlogik erster Stufe angesiedelt,[454] weil bspw. nicht mehr als zweistellige Prädikate erlaubt werden, die Prädikatenlogik erster Stufe jedoch hinsichtlich der Stelligkeit der Prädikate keine Einschränkungen vorsieht.[455] Mittels Beschreibungslogiken entwickelte Sprachen und hiermit repräsentierte Ontologien haben üblicherweise gute rechenbetonte (*computational*) Eigenschaften, sie sind entscheid- und begründbar und oftmals in angemessener, d. h. für den Benutzer vertretbarer Zeit mit einer Inferenzmaschine zu durchlaufen.[456]

In einer *Rahmenlogik* werden die Elemente der Prädikatenlogik erster Stufe mit Konstrukten aus der objektorientierten Programmierung verbunden. Dabei stellen im Gegensatz zur Prädikatenlogik, wo die Prädikate die zentralen Modellierungsprimitive sind, Klassen die zentralen Modellierungsprimitive (d. h. Frames) dar. Auf syntaktischer Ebene stellt das Kalkül der Rahmenlogik eine Obermenge der Prädikatenlogik erster Stufe dar.[457] Hinsichtlich seiner Ausdrucksmächtigkeit wird die Rahmenlogik zwischen den zwei Stufen der Prädikatenlogik angesiedelt.[458]

In der *Prädikatenlogik zweiter Stufe* wird eine größere Ausdrucksmächtigkeit als in den bisher vorgestellten Kalkülen erreicht, indem Quantifizierungen auch über Prädikaten- und Funktionssymbole erlaubt werden.[459]

Die Ausdrucksmächtigkeit einer Wissensrepräsentationssprache verhält sich reziprok zum Aufwand, der betrieben werden muss, um eine Aussage in der betreffenden Logik zu erhalten. Dies betrifft in diesem Zusammenhang auch die Anforderungen, die an eine Inferenzmaschine

---

453) Auf weitere Erläuterungen wird an dieser Stelle verzichtet, weil sie nicht weiter der Argumentation dieser Arbeit dienen. Der interessierte Leser sei als Einstieg verwiesen auf Baader, Horrocks et al. (2004).

454) Vgl. Borgida (1996). Hier wird deutlich, dass eine herkömmliche Beschreibungslogik eine echte Teilmenge der Prädikatenlogik erster Stufe darstellt.

455) Des Weiteren kann ein Konzept nicht zugleich Instanz eines anderen Konzepts sein. Diese Modellierung ist bspw. jedoch innerhalb der Rahmenlogik zulässig, die in dieser Arbeit auch aus diesem Grund „oberhalb" der Prädikatenlogik erster Stufe angesiedelt wird.

456) Vgl. Decker (2002), S. 81.

457) Vgl. Decker (2002), S. 83.

458) Vgl. Kifer, Lausen et al. (1995), S. 742 f.

459) Vgl. Schöning (2000), S. 56. Eine nähere Beschreibung an dieser Stelle erscheint nicht notwendig, weil keine der ermittelten Sprachen zur Repräsentation von Ontologien eine derartige Ausdrucksmächtigkeit vorweisen kann. Da es schon keinen Theorem-Beweiser gibt, der für die Prädikatenlogik erster Stufe terminiert, gibt es auch keinen, der für die zweite Stufe terminiert.

gestellt werden.[460] Je ausdrucksmächtiger ein Kalkül ist, desto häufiger kommt es zu Unentscheidbarkeiten oder zu für den Benutzer nicht mehr annehmbaren Laufzeiten bei den Schlussfolgerungen aufgrund eines exponentiell anwachsenden Ressourcenbedarfs zum Beweis der Erfüllbarkeit, Allgemeingültigkeit oder Widersprüchlichkeit von Formeln, sofern es sich um entscheidbare Formeln handelt.

4.2.1.2.3 Traditionelle versus webbasierte Verwendung

Anhand der vorgestellten Kalküle werden aus der Literatur bekannte Sprachen zur Spezifikation von Ontologien gegliedert. Hierzu lässt sich anfänglich in der Verwendung von Repräsentationssprachen eine Unterscheidung von „traditionellen" Sprachen versus Markup-Sprachen zur Verwendung für das Semantic Web vornehmen.[461] Auf diese Einteilung wurde bereits im Kapitel zu den Einsatzgebieten von Ontologien zurückgegriffen.

4.2.2 Untersuchte Repräsentationssprachen

4.2.2.1 Traditionelle Repräsentationssprachen

Als traditionelle Sprachen werden an dieser Stelle Prolog, Ontolingua, LOOM, CycL, F-Logic und OCML kurz vorgestellt.[462]

460) Vgl. Strang (2003), S.6.

461) Der Verfasser folgt damit einer Unterscheidung, die ebenfalls in der Literatur vorgenommen wird, bspw. von Gómez-Pérez, Fernández-López et al. (2004), S. 199 ff. Fensel folgt ebenfalls dieser Gliederung, indem er zunächst RDF und XML vorstellt, anschließend in prädikatenlogische, rahmenbasierte und beschreibungslogische Ansätze kurz unterscheidet und schließlich in RDF und XML darstellt, wie Eigenschaften dieser Ansätze für das Semantic Web einsatzfähig gemacht werden können (vgl. Fensel (2004), S. 11 ff.).

462) In der Literatur finden sich wenige weitere traditionelle Sprachen zur Repräsentation von Ontologien. Diese wurden aus vier Gründen ausgegrenzt, die jeweils bei Nichterfüllung zum Ausschluss der Sprache führten: 1. Die Repräsentationssprache erlaubt nicht die Formulierung von Regeln. 2. Es existiert keine Inferenzmaschine, die eine Anwendung der Ontologie zwecks Explizierung impliziten Wissens in der jeweiligen Sprache erlaubt. 3. Die Sprache gilt allgemein als veraltet. 4. Die Sprache dient lediglich als „Mediator", d. h. die Sprache dient bspw. entweder als Standard zur Vereinheitlichung oder als Übersetzungsprotokoll. Beispiele hierfür sind KIF bzw. OKBC.
Das Knowledge Interchange Format (KIF) wurde zum Austausch von *Wissen* zwischen ungleichartigen Programmen entwickelt. Es enthält eine deklarative Semantik (die Bedeutung der Ausdrücke in der Repräsentation wird verstanden, ohne zuvor auf einen Interpreter zurückgreifen zu müssen). KIF beinhaltet Elemente der Prädikatenlogik erster Ordnung, es unterstützt die Repräsentation von Meta-Wissen und die Definition von Objekten, Funktionen und Relationen. Die Sprache soll eine Übersetzungsfunktion erfüllen und somit als Mediator zwischen anderen Sprachen eingesetzt werden. Die Sprachbeschreibung enthält sowohl eine Spezifikation für die Syntax als auch für die Semantik. Der KIF-Kern ähnelt sehr stark F-Logic (für weitere Informationen siehe: http://www-ksl.stanford.edu/knowledge-sharing/kif, http://logic.stanford.edu/kif/kif.html, Zugriff am 18.03.2007).
Die *Open Knowledge Base Connectivity* (OKBC) wurde als rahmenbasiertes Protokoll entwickelt, um einheitlich auf die Wissensbasen unterschiedlicher Wissensbasierter Systeme zugreifen zu können (vgl. Chaudhri, Farquhar et al. (1998) und http://www.ai.sri.com/~okbc/okbc-2-0-3.pdf, Zugriff am 18.03.2007).

4.2.2.1.1 Prolog

Die Sprache ***Pro****gramming in* ***Log****ic* (Prolog) basiert auf Hornklauseln, die eine Untermenge der Prädikatenlogik erster Stufe darstellen. Hornklauseln beziehen sich auf die Eigenschaft, dass der logische Operator „Adjunktion" nicht in den Formeln der Konklusion einer Implikation vorkommen darf.[463] Prolog-Programme bestehen aus Fakten und Regeln, die gemeinsam die Wissensbasis bilden. Der Benutzer formuliert Anfragen an diese Wissensbasis. Mit Hilfe eines Interpreters werden die Fakten und Regeln genutzt, um systematisch eine Antwort zu finden. Ein positives Resultat bedeutet, dass die Antwort logisch aus der Wissensbasis ableitbar ist. Ein negatives Resultat ermöglicht lediglich die Erkenntnis, dass aufgrund der vorliegenden Wissensbasis keine Antwort ermittelt werden kann.

4.2.2.1.2 Ontolingua

Basierend auf KIF wurde 1992 Ontolingua[464] vom Knowledge Systems Lab (KSL) der Universität Stanford vorgestellt. Aufgrund seiner Anlehnung an die Rahmenlogik unterstützt Ontolingua die Spezifikation von Konzepten, Taxonomien über Konzepte, Relationen, Funktionen und Regeln. Die rahmenbasierten Modellierungsprimitive in Ontolingua werden über Regeln in der Sprache festgelegt. Die große Ausdrucksstärke führte zu Problemen hinsichtlich der Erstellung von Schlussfolgerungsmechanismen.[465] Auch ein entwickelter Theorem-Beweiser für KIF-Ausdrücke konnte hieran wenig ändern.[466]

4.2.2.1.3 LOOM

LOOM[467] basiert auf einem Kalkül, das den Beschreibungslogiken zugerechnet wird. Zusätzlich basiert LOOM auf Produktionsregeln. LOOM wurde von 1986 bis 1995 am Institut für Informationssysteme der Universität South California entwickelt und gilt als eine sehr ausdrucksstarke Sprache, weil es bspw. die automatische Klassifikation von Konzepten erlaubt. Ursprünglich wurde LOOM – ähnlich wie Prolog – nicht entwickelt, um Ontologien darzustellen, sondern um Expertensysteme zu entwickeln.[468] Ihre Ausdrucksmächtigkeit ermöglicht jedoch auch die Spezifizierung von Ontologien. Erlaubt werden die Repräsentation von Konzepten, Taxonomien über Konzepte, Relationen, Funktionen und Regeln.

463) Vgl. Sowa (2000), S. 19.

464) Vgl. http://ksl.stanford.edu/software/ontolingua, Zugriff am 18.03.2007.

465) Vgl. Corcho, Fernández-López et al. (2003), S. 54.

466) Vgl. Gómez-Pérez, Fernández-López et al. (2002), S. 75.

467) Vgl. Loom Guide (1991). LOOM ist sowohl eine Repräsentationssprache als auch eine Softwareumgebung zur Verwendung in Wissensbasierten Systemen.

468) Vgl. Gómez-Pérez, Fernández-López et al. (2004), S. 216.

4.2.2.1.4 CycL

CycL[469] ist eine formale Sprache, deren Syntax und Semantik sich ursprünglich aus Teilen der Prädikatenlogik erster Stufe ergab. CycL unterscheidet bei Termen zwischen semantischen Konstanten, nicht-atomaren Termen (NATs), Variablen, Nummern, Strings etc. Terme werden kombiniert zu CycL-Ausdrücken, die letztlich CycL-Sätze bilden, die in ihrer Bedeutung "geschlossen" sind, d. h. keine freien Variablen mehr enthalten.[470] Eine Menge von CycL-Sätzen bildet eine Wissensbasis. Mittlerweile, d. h. aufgrund der kontinuierlichen Weiterentwicklung von CycL, gehen Syntax und Semantik von CycL über die erste Stufe der Prädikatenlogik hinaus.[471] Bspw. unterscheidet CycL vier Existenzquantoren (thereExists, thereExistAtLeast, thereExistAtMost und thereExistExactly) und erlaubt die Quantifizierung über Prädikate und Funktionen.[472]

4.2.2.1.5 F-Logic

F-Logic ist eine Sprache zur Repräsentation von Wissen über Konzepte und ihre Relationen. Die Spezifikation von F-Logic entstammt der Forschung aus dem Themengebiet der deduktiven und objektorientierten Datenbanken.[473] F-Logic stellt eine Kombination der Repräsentationsarten Frames[474] und Prädikatenlogik dar.[475] Es kann in einer Faktenbasis Wissen über die Domäne abgelegt werden, anhand derer mit Regeln „neues" Wissen expliziert werden kann. Es lässt sich bereits bei einigen Sprachen zu den Kalkülen der Beschreibungslogik ein Objektkonzept erkennen, jedoch wird in F-Logic die Objektorientierung im Gegensatz zu den auf Beschreibungslogik basierenden Sprachen oder Ontolingua nicht nachträglich simuliert, sondern von Anfang an bei der Konzeption berücksichtigt, d. h. sie wurde über eine explizite Definition der Semantik festgelegt.

4.2.2.1.6 OCML

Die Operational Conceptual Modeling Language (OCML) wurde am Knowledge Media Institute in Großbritannien entwickelt.[476] OCML kann als eine Art operationalisierte Ontolingua verstanden werden, die Theorem-Beweisen ermöglicht.[477] Sie zählt zu den framebasierten

---

469) Vgl. http://www.cyc.com/cycdoc/ref/cycl-syntax.html, Zugriff am 18.03.2007.

470) Vgl. Fensel (2004), S. 22.

471) Vgl. Lenat (1995), S. 35.

472) Vgl. Fensel (2004), S. 22.

473) Vgl. Gómez-Pérez, Fernández-López et al. (2004), S. 231. Siehe Kifer, Lausen et al. (1995) für eine Gesamtabhandlung zu F-Logic.

474) Siehe Kapitel 4.1.3.2.5, Seite 84.

475) Vgl. Decker (1998), S. 9.1 ff.

476) Vgl. Motta (1999), S. 47 ff.

477) Vgl. Gómez-Pérez, Fernández-López et al. (2004), S. 227.

Ansätzen mit einer Syntax ähnlich wie LISP.[478] Es können Konzepte, Relationen, Funktionen und Regeln ausgedrückt werden.

4.2.2.2 Repräsentationssprachen für das Semantic Web

Als Spezifikationssprachen für das Semantic Web werden die derzeit am weitesten verbreiteten Sprachen berücksichtigt. An dieser Stelle wird auf ausdatierte Vorgängersprachen nicht weiter eingegangen (wie bspw. OIL, das in DAML+OIL aufgegangen ist). Aufgrund der Standardisierungsbemühungen des W3C (World Wide Web Consortium) – ohne dessen Unterstützung eine breite Akzeptanz und Verwendung einer Sprache für das Internet nicht zu erwarten ist – finden sich hier nur Sprachen wieder, die mit einer eigenen Arbeitsgruppe im W3C vertreten sind. Hierzu zählen XML, RDF(S), DAML+OIL, OWL und RuleML.

4.2.2.2.1 XML

XML (eXtensible Markup Language) ist aufgrund von Vereinfachungsbemühungen aus SGML (Standard Generalized Markup Language) hervorgegangen und vom W3C als Standard empfohlen worden.[479] Sie stellt eine Meta-Sprache dar, die insbesondere den Erfordernissen des WWW Rechnung trägt. SGML (und somit auch XML) dienen als Meta-Sprachen zur Definition von Markup-Sprachen für die Erstellung und Verarbeitung von Dokumenten im WWW. Es besteht die Möglichkeit, beliebige Tags (Dokumenttypdefinitionen) festzulegen, die anschließend einem Agenten ermöglichen, die Semantik einzelner Dokumentpassagen zu erfassen und weiterzuverarbeiten. Ein in einem XML-Dokument enthaltenes Element muss in einer Document Type Definition (DTD) aufgeführt und definiert werden. Die DTD kann dabei sowohl intern als auch extern zum Dokument vorgehalten sein. XML selbst kann als Meta-Sprache genutzt werden, um Markup-Sprachen für die Spezifikation von Ontologien zu entwickeln. Eine bekannte Anwendung von XML stellt RDF dar, das im folgenden Kapitel beschrieben wird.

4.2.2.2.2 RDF(S)

Resource Description Framework (Schema) (RDF(S))[480] ist eine weitere Entwicklung und Standardisierung des W3C. Ziel der Entwicklung war es, die unterschiedlichen Standards hinsichtlich der Syntax von Meta-Daten und möglichen Beschreibungssprachen für Schemata auf dem Weg von der Maschinenlesbarkeit zur Maschinenverstehbarkeit zu vereinen. RDF enthält eine Syntax-Spezifikation (RDF) und eine Schema-Spezifikation (RDF-S). RDF besitzt seit 1999 den Status einer W3C-Recommendation, während RDF-S erst seit 2004 als Recommendation eingestuft wird. Zusammengenommen bildet RDF(S) eine Infrastruktur zu Kodierung, Austausch und Wiederverwendung von strukturierten Meta-Daten. Insbesondere

478) Zu LISP siehe Graham (1997).

479) Vgl. http://w3.org/XML/, für Tutorien siehe: http://www.programmingtutorials.com/xml.aspx, Zugriff am 18.3.2007. Eine kurze Einführung zu XML bietet Eckstein (2000).

480) Vgl. http://w3.org/RDF/, Zugriff am 18.03.2007.

wird dies genutzt für Suchmaschinen, intelligente Agenten, Informationsbroker, Browser und nicht zuletzt für menschliche Nutzer. Es bietet die Möglichkeit, semantische Informationen (bspw. zur Beschreibung von Klassen) maschinell zu verarbeiten. Das RDF-Schema erlaubt Entwicklern, Klassen (*Classes*) von Ressourcentypen (*Resource Types*) und Eigenschaften (*Properties*) zu spezifizieren, um Beschreibungen dieser Klassen, Beziehungen zwischen Eigenschaften und Klassen und Einschränkungen (*Constraints*) bezüglich erlaubter Kombinationen von Klassen, Eigenschaften und feststehenden Werten von Eigenschaften zu übermitteln.[481] Gerade in dieser Limitierung liegen die Schwächen von RDF(S), weil sich lediglich Konzepte, Taxonomien von Konzepten und zweistellige Relationen darstellen lassen.[482] Gemäß der Definition dieser Arbeit für Ontologien lassen sich mit RDF(S) keine Ontologien abbilden, weil nicht die Möglichkeit besteht, Regeln zu implementieren.[483]

4.2.2.2.3 DAML+OIL

DAML+OIL[484] stellt eine spezielle semantische Markup-Sprache für Ressourcen des World Wide Webs dar. DAML+OIL ist eine gemeinsame Entwicklung europäischer und amerikanischer Forschungseinrichtungen insbesondere für das Semantic Web, das in jüngster Zeit in das Interesse der informationstechnischen Forschung gerückt ist.

DAML+OIL baut auf RDF(S) auf und erweitert RDF(S) um weitere Modellierungsmöglichkeiten. Bspw. wird es möglich anzugeben, dass zwei Klassen sich disjunkt zueinander verhalten (daml:disjointWith), und es können Kardinalitäten für Eigenschaften angegeben werden (daml:maxCardinality; daml:minCardinality). Die enthaltenen Modellvorgaben finden vornehmlich in framebasierten und Description-Logics-Sprachen Verwendung. DAML+OIL erlaubt die Repräsentation von Konzepten, Taxonomien von Konzepten, binären Relationen und Funktionen.

4.2.2.2.4 OWL

Um die beschränkte Ausdrucksmächtigkeit von RDF(S) zu erweitern, wurde die Web Ontology Language (OWL), unterstützt vom W3C, entwickelt.[485] Aufbauend auf DAML+OIL[486]

---

481) Vgl. http://www.w3.org/TR/rdf-schema/, Zugriff am 18.03.2007.

482) Vgl. Corcho, Fernández-López et al. (2003), S. 55.

483) Allenfalls im Sinne von *leichtgewichtigen Ontologien* (siehe Kapitel 4.1.1) lassen sich mit RDF(S) Ontologien spezifizieren. Diese wären dann allerdings als noch leichtgewichtiger einzustufen als „herkömmliche" leichtgewichtige Ontologien, die bspw. mit OWL Lite (siehe Kapitel 4.2.2.2.4) spezifiziert worden sind. So kann die Disjunktheit zweier Klassen mittels OWL Lite im Gegensatz zu RDF(S) festgelegt werden (weitere Beispiele sind: Festlegung von Kardinalitäten, Boolesche Kombination von Klassen, Zuordnung bestimmter Eigenschaften auf Klassen-Ebene).

484) Vgl. http://www.w3.org/TR/daml+oil-reference, Zugriff am 18.3.2007.

485) Vgl. Antoniou, van Harmelen (2004), S. 67. Siehe zu RDF(S) und dessen Limitierungen auch Kapitel 4.2.2.2.2, S. 107 f.

486) Zu den Abweichungen bei den Modellierungsprimitiven von OWL gegenüber denen von DAML+OIL siehe Gómez-Pérez, Fernández-López et al. (2004), S. 274 f.

und RDF(S)[487] stellt OWL die jüngste Sprache im Streben nach einem „verstehenden" WWW, einem Internet auf der Basis semantischer Technologien dar und besitzt seit kurzer Zeit (Februar 2004) den Status einer Empfehlung durch das W3C.[488] Es wird damit gerechnet, dass sich die Empfehlung seitens des W3C positiv auf Akzeptanz und Verwendung der Spezifikationssprache auswirken wird.

Grundsätzlich lässt sich OWL in drei Subsprachen unterteilen, die mit einer zunehmenden Ausdrucksmächtigkeit versehen wurden:[489]

- *OWL Lite* ist eine einfache Sprache, die hauptsächlich Hierarchien von Konzepten ausdrücken kann und einfache Beschränkungseigenschaften (*Constraints*) ermöglicht.[490] OWL Lite dient vornehmlich der schnellen Integration bereits existierender Klassifikationssystematiken (bspw. Thesauri und Taxonomien).

- *OWL DL* (DL steht als Kürzel für *Description Logic*) umfasst die vollständigen OWL-Modellierungsprimitive[491], die jedoch nur mittels einiger Beschränkungen verwendet und interpretiert werden können.[492] Dies dient dem Ziel, dass ein OWL DL-Kode noch in endlicher Zeit hinsichtlich der Schlussfolgerungen ausgeführt werden kann. Die Beschränkungen gehen jedoch zu Lasten einer vollständigen Kompatibilität zu RDF(S).[493]

- *OWL Full* verwendet sämtliche Modellierungsprimitive der Sprache. Es ermöglicht eine umfassende Interpretation der vollständigen OWL-Modellierungsprimitive entsprechend der Möglichkeiten, die durch RDF determiniert werden. Der Vorteil von OWL Full liegt in der vollständigen Aufwärtskompatibilität zu RDF: jedes zulässige RDF-Dokument entspricht einem zulässigen OWL-Dokument.[494] Hierin liegt auch der Nachteil von OWL Full: die Ausdrucksmächtigkeit ist so groß, dass es keine Möglichkeit gibt, jede OWL-Full-Ontologie hinsichtlich ihrer Schlussfolgerungsmöglichkeiten überhaupt computergestützt zu verwenden.[495]

---

487) In OWL werden die Modellierungsprimitive von RDF(S) weiterverwendet, ohne eine Umbenennung dieser Modellierungsprimitive vorzunehmen (bspw. *rdfs:subClassof*, *rdfs:subPropertyOf*).

488) Für ausführlichere Informationen zu OWL und den Bestrebungen zu einem „Semantic Web" siehe auch: http://www.w3.org/2004/OWL/, Zugriff am 18.03.2007.

489) Vgl. McGuinness, van Harmelen (2004), Kapitel 1.3.

490) So erlaubt OWL Lite bspw. lediglich die Festlegung der Kardinalitäten 0 und 1.

491) Eine sehr übersichtliche Darstellung der Modellierungsprimitive von OWL DL und OWL Lite findet sich bei Gómez-Pérez, Fernández-López et al. (2004), S. 66.

492) Bspw. ist es möglich, in OWL Full ein Konzept sowohl als Unterkonzept als auch gleichzeitig als Instanz eines anderen Konzepts zu verwenden. Diese Möglichkeit der Modellierung wird durch die Primitive von Description Logic in OWL DL untersagt.

493) Wie bereits in der vorangehenden Fn. erläutert, kann ein Konzept nicht zugleich Instanz sein. Diese Modellierung wäre jedoch in RDF(S) zulässig. Vgl. zur Problematik der Kompatibilität von RDF(S) und OWL Horrocks, Patel-Schneider et al. (2003), S. 14 ff., und insbesondere S. 22 zur Einschränkung von OWL DL gegenüber RDF(S).

494) Vgl. Antoniou, van Harmelen (2004), S. 70.

495) Vgl. McGuinness, van Harmelen (2004), Kapitel 1.3.

Die offizielle Austausch-Syntax für OWL ist RDF/XML.[496]

4.2.2.2.5 Rule Markup Language

Mit OWL lassen sich lediglich leichtgewichtige Ontologien spezifizieren. Um auch schwergewichtige Ontologien spezifizieren zu können, wird es notwendig, Regeln formulieren zu können. Unter der Bezeichnung RuleML[497] (Rule Markup Language) wird zur Zeit versucht, einen Standard zu entwickeln, der OWL um die Möglichkeit des Schlussfolgens mittels Regeln erweitert.

Vergleichbar mit OWL soll RuleML in aufeinander aufbauende Untersprachen gegliedert werden. Aktuell liegt dem W3C zur Begutachtung ein Vorschlag für den Dialekt SWRL (SemanticWeb Rule Language) vor, der OWL Lite und OWL DL mit einer Untergruppe von Modellierungsprimitiven der RuleML kombiniert.[498]

4.2.3 Auswahl einer Repräsentationssprache

4.2.3.1 Betrachtung der zentralen Anforderungen

Tabelle 3, S. 113, fasst die ermittelten Spezifikationssprachen, die in dieser Arbeit betrachtet wurden, und deren wichtigste Eigenschaften synoptisch zusammen.[499]

Die Spaltenüberschriften enthalten hierzu die zentralen Anforderungen an eine Sprache zur Wissensrepräsentation mittels Ontologien aus Kapitel 4.2.1.2.1, S. 98 f. Auf die Anforderung *Computerverarbeitbarkeit* wird verzichtet, weil diese Anforderung von allen untersuchten Sprachen erfüllt wird.[500]

In den ersten beiden Spalten der Tabelle wird die Ausdrucksmächtigkeit einer Repräsentationssprache hinsichtlich der Darstellung einer Ontologie untersucht. Es wird die Skala (*niedrig*, *mitte*l, *hoch* und *sehr hoch*) von S. 98 zur *Ausdrucksmächtigkeit hinsichtlich der Darstellung ohne Regeln* (im Sinne einer leichtgewichtigen Ontologie) verwendet. In der zweiten Spalte wird die Ausdrucksmächtigkeit zur *Darstellung von Regeln* untersucht.

---

496) Vgl. Patel-Schneider, Hayes et al. (2004), Kapitel 1.

497) Vgl. hierzu auch http://www.ruleml.org/ und http://www.w3.org/Submission/SWRL/, Zugriff am 14.03.2007.

498) Vgl. Horrocks, Patel-Schneider et al. (2004).

499) Dabei wird jedoch die RuleML außer Acht gelassen, weil sie im eigentlichen Sinne keine eigenständige Sprache zur Repräsentation von Ontologien darstellt.
In der Literatur finden sich einige wenige Ansätze zur Evaluierung von Sprachen zur Repräsentation von Ontologien (vgl. Corcho, Fernández-López et al. (2003); S. 54 ff.; Corcho, Gómez-Pérez (2000); Gómez-Pérez, Fernández-López et al. (2004), S. 202 ff. Letztlich betonen diese Untersuchungen zumeist jedoch die Abhängigkeit der Verwendung einer Sprache von dem Einsatzzweck der Ontologie.

500) Dieser Umstand erscheint nicht weiter verwunderlich, weil gerade die Problemstellung der Arbeit eine Computerverarbeitbarkeit vorgibt, so dass es wenig Sinn gemacht hätte, nicht computerverarbeitbare Sprachen zu betrachten.

Die *Nachvollziehbarkeit* wird in der dritten Spalte der Tabelle untersucht. Dabei wird eine Skala mit den Ausprägungen *unübersichtlich*, *mittel*, *hoch* und *sehr hoch* für die einzelnen Sprachen angewendet. Die Ausprägung *unübersichtlich* wird vergeben, wenn eine Sprache im Gegensatz zu einer prädikatenlogischen Formulierung übermäßig viele Worte für die Darstellung einer Aussage benötigt. Je näher sich eine Darstellung einer Aussage innerhalb einer Sprache an die prädikatenlogisch äquivalente Darstellung annähert, desto mehr wird eine „bessere" Ausprägung vergeben.[501]

Die beiden verbleibenden Spalten dienen an dieser Stelle vorwiegend dem weiteren Verständnis.[502] In der vierten Spalte wird eine gegebene verteilte Anwendung im Sinne des Semantic Webs untersucht. Die letzte Spalte untersucht, ob die Repräsentationssprache originär für die Erstellung (und damit für den Bedarf) von Ontologien entwickelt wurde.[503]

4.2.3.2 Begründung der Auswahl

Die Markup-Sprachen bauen aufeinander auf oder sind gegenseitig übersetzbar. Dabei gilt OWL Full zurzeit als die ausdrucksmächtigste Sprache. Derzeit sind diese Markup-Sprachen jedoch nicht in der Lage, Regeln zu spezifizieren, wie sie mit dem Ontologieverständnis dieser Arbeit gefordert werden.[504] Ferner erscheinen Markup-Sprachen nicht notwendig für die FMEA-Entwicklung, weil für das hier vorgestellte Wissensbasierte System die gemeinsame Verwendung einer Wissensbasis im weiteren Sinne über einen Server geplant wird. Ein Grund hierfür liegt in der einfacheren Sicherstellung einer gemeinsamen Verwendung der Modellierung einer Domäne. Wenn lediglich eine nicht verteilte Anwendung ermöglicht wird, so bleibt den Benutzern lediglich der Zugriff auf diese eine vorhandene Modellierung. Die Möglichkeiten der Inselschaffungen werden unterbunden. Ein weiterer Vorteil, der sich mit der Verwendung eines solitären Modells ergibt, wird in der Möglichkeit der vereinfachten Wahrung von betrieblichem Know-how gesehen. Gerade die angestrebte einfache Nachvollziehbarkeit einer Wissensrepräsentation in Form einer Ontologie muss besonders gesichert werden, weil sich aus der einfachen Nachvollziehbarkeit auch ein größeres Potential des Missbrauchs ergibt.[505] Hier wird unterstellt, dass eine an einem Ort verwendete Ontologie tendenziell leichter vor Missbrauch zu schützen ist als eine an mehreren Orten verteilte Ontologie, die gerade mit den Vorteilen des Semantic Webs angestrebt wird. Bei den traditionellen

---

501) Zusätzlich werden auch die Möglichkeiten von Frames und ihrer Kapselung von Wissen in die Betrachtungen einbezogen, sofern diese eine leichtere Nachvollziehbarkeit ermöglichen.

502) Siehe hierzu auch die Fn. 439, S. 99.

503) Hier ließe sich tendenziell unterstellen, dass originär entwickelte Sprachen eher die spezifischen Anforderungen an die Darstellung von Ontologien erfüllen als Sprachen, die ursprünglich für einen anderen Zweck entwickelt wurden. Wie jedoch mehrfach erwähnt wurde, dient diese Anforderung lediglich der Verdeutlichung und wurde für die Auswahl der hier verwendeten Repräsentationssprache nicht berücksichtigt.

504) Stuckenschmidt und van Harmelen geben eine kurze Analyse hinsichtlich der Gemeinsamkeiten und Unterschiede von OWL und F-Logic. Sie kommen dabei zu dem Schluss, dass es sich bei F-Logic um die ausdrucksmächtigere Sprache handelt, gerade weil sie die Modellierung von Regeln erlaubt (vgl. Stuckenschmidt, van Harmelen (2005), S. 188 f.).

505) Der Grund hierfür liegt vornehmlich in der erleichterten Wiederverwendbarkeit des repräsentierten Wissens.

Sprachen zur Ontologie-Darstellung verfügen LOOM und Ontolingua über die größte Ausdrucksmächtigkeit.

Für eine Verwendung von F-Logic in dieser Arbeit sprechen neben der Forderung, Regeln implementieren zu können, dass F-Logic als einzige Sprache über die Möglichkeit der Behandlung von Ausnahmen verfügt (d. h. eine nicht monotone Vererbung wird ermöglicht) und ihre Beweismethoden als einzige als korrekt und vollständig (sound and complete) anerkannt werden.[506] Das „ausschlaggebendste" Argument für die Verwendung von F-Logic war jedoch die einfache Nachvollziehbarkeit der Kodierung, die als einzige mit dem Merkmalswert *sehr hoch* versehen wurde.[507] Im Gegensatz hierzu erhielten die Markup-Sprachen durchweg den Merkmalswert *unübersichtlich* aufgrund ihrer auch als „wortreich"[508] genannten Darstellungsweise, die damit verbunden ist, übermäßig viel Kodierung zu produzieren, um einen Sachverhalt festzulegen.

Ein Großteil der Modellierungsprimitive von F-Logic kann übersetzt werden in Markup-Sprachen. Für den Einsatzzweck der FMEA-Ontologie-Erstellung in dieser Arbeit und die Darstellung der Ergebnisse offerierte F-Logic die „passendsten" Modellierungsprimitive.

---

506) Vgl. Corcho, Gómez-Pérez (2000), S. 3.8.

507) Vgl. Friedland, Allen (2003), S. 14.

508) Vgl. Horrocks, Patel-Schneider et al. (2003), S. 17.

Sprache	Ausdrucksmächtigkeit hinsichtlich der Darstellung ohne Regeln	Darstellung von Regeln	Nachvollziehbarkeit	verteilte Anwend-barkeit	originär Onto-logie-entwicklung
Prolog	sehr hoch	ja	hoch	nein	nein
Ontolingua	sehr hoch	ja	mittel	nein	ja
CycL	sehr hoch	ja	mittel	nein	ja
LOOM	sehr hoch	ja	mittel	nein	nein
F-Logic	sehr hoch	ja	sehr hoch	nein	nein
OCML	hoch	ja	mittel	nein	ja
XML	hoch	nein	unübersichtlich (geschwätzig)	ja	nein
RDF(S)	mittel	nein	unübersichtlich (geschwätzig)	ja	nein
DAML+OIL	mittel/hoch	nein	unübersichtlich (geschwätzig)	ja	ja
OWL	abhängig vom gewähl-ten Umfang (Light, Full, DL) mittel bis sehr hoch	nein	unübersichtlich (geschwätzig)	ja	ja

Tabelle 3: Synopse Spezifikationssprachen

4.3 Explikation von Wissen in einer Ontologie

Um später mit der Hilfe einer Repräsentationssprache eine FMEA-Ontologie aufzustellen, wird an dieser Stelle kurz auf den formalen Aufbau einer Ontologie eingegangen. Die hier vorgestellten (formalen) Modellierungsprimitive einer Ontologie werden in Kapitel 7, S. 219 ff., genutzt, um in der ausgewählten Repräsentationssprache (F-Logic) eine FMEA-Ontologie zu entwickeln. In diesem Sinne bedeutet die Repräsentation des Wissens einer Ontologie die Anwendung des Meta-Wissens, das in einer formalen Definition zu Ontologien enthalten ist.

In dieser Arbeit umfassen die Modellierungsprimitive einer (schwergewichtigen) Ontologie ein Tupel O über einem Universum U :[509]

$$O = \langle C, R, F \rangle$$

bestehend aus:

- $C \subseteq U$ umfasst die Menge der Konzepte.
- $R \subseteq U$ umfasst die Menge der Relationen mit $R = R^H \cup R^R$ und $R^H \cap R^R = \varnothing$. R^H beinhaltet die hierarchischen terminologischen Relationen (Hierarchierelationen) zwischen Konzepten und R^R beinhaltet die verbleibenden denkbaren Relationen zwischen Konzepten. Dabei gilt $R^R \subseteq C_1 \times ... \times C_n$ und $R^H \subseteq C \times C$. Die hierarchischen Relationen haben eine besondere Relevanz in diesem Zusammenhang gegenüber den verbleibenden Relationen, weil die kognitiven Fähigkeiten des menschlichen Abstraktionsvermögens dahin tendieren, mittels hierarchischer terminologischer Relationen Konzepte zu strukturieren.[510] Insbesondere im Kontext der Arbeit (FMEA-Erstellung) werden hierarchische terminologische Relationen genutzt, um Konzepte für Produkte oder Prozesse zu systematisieren.
  Hierarchierelationen lassen sich in Abstraktionsrelation und Bestandsrelation unterteilen.[511] Die Abstraktionsrelation (auch generische Relation genannt) stellt eine hierarchische terminologische Relation dar, bei der der Konzeptinhalt des untergeordneten Konzepts den Konzeptinhalt des übergeordneten Konzepts einschließt, wobei sich das untergeordnete Konzept in mindestens einem zusätzlichen Merkmal vom übergeordneten Konzept unterscheidet (z. B. kann das Oberkonzept `kraftfahrzeug` in die Unterkonzepte `personenkraftfahrzeug` und `lastkraftfahrzeug` unter-

---

509) Für ähnliche, jedoch abweichende formale Definitionen vgl. Erdmann (2001), S. 76 und Maedche, Staab (2001), S. 74. Diese formale Darstellung deckt sich mit dem Kern der Definition von Ontologien in dieser Arbeit aus Kapitel 2.1.2.4, S. 28 f.
Die Existenz eines Universums U wird hier als Axiom gefordert (Universenaxiom). Das Universum U umfasst beliebige unbekannte und bekannte Objekte sowie Mengen von Objekten.

510) Vgl. Johnson-Laird (1983), S. 204; Strube, Habel et al. (2003), S. 28. ff. Andersherum weist Bibel in diesem Zusammenhang darauf hin, dass sich kognitive Vorteile aus einer solchen Darstellung ergeben, weil solche Beziehungen vom Menschen in grafischer Form schneller abgelesen werden können (vgl. Bibel (1993), S. 48). Zur Rekonstruktion mentaler Modelle mit Hilfe Wissensbasierter Systeme vgl. Opwis (1992).

511) Vgl. DIN 1463-1:1987, S. 5 f.

schieden werden).[512]

Die Bestandsrelation (auch Teil-Ganzes-Relation oder partitive Relation genannt) stellt eine hierarchische terminologische Relation dar, bei der sich die untergeordneten Konzepte auf die Teile eines Gegenstands beziehen und sich das übergeordnete Konzept auf einen Gegenstand als Ganzes bezieht (z. B. können einem Konzept `schraubmaschine` die beiden Konzepte `bedieneinheit` und `transporteinheit` mittels der Bestandrelation als untergeordnete Konzepte zugeordnet werden).[513]

- Die Menge $F = F_{fix} \cup F_{spec}$ mit $F_{fix} \cap F_{spec} = \emptyset$ beinhaltet unbeschränkte prädikatenlogische Formeln als Regeln, die zusammengesetzt werden aus C, R und logischen Partikeln, die sich in der Prädikatenlogik 1. Stufe finden (bspw. Negation $\neg$, Subjunktion $\rightarrow$ und Konjunktion $\wedge$). F_{fix} beinhaltet dabei die Menge der Regeln, die Eigenschaften von Prädikatssymbolen reflektieren (wie Attribute und Integritätsregeln). Diese Regeln reflektieren die Semantik vordefinierter Prädikatssymbole so präzise als möglich. F_{spec} beinhaltet Regeln, die durch die spezifische Domäne der zu definierenden Ontologie restringiert werden (Inferenzregeln). Diese spezifischen Regeln werden vornehmlich genutzt, um implizites Wissen der Wissensbasis zu explizieren.

512) Vgl. DIN E 2342:2004, Abschnitt 4.6.5.1.1, S. 7. Dabei stellt ein Merkmal eine durch Abstraktion gewonnene Denkeinheit dar, die eine Eigenschaft von Gegenständen repräsentiert, die zur Konzeptbildung und Konzeptabgrenzung verwendet wird (vgl. ebenda, Abschnitt 4.2, S. 5).

513) Vgl. DIN E 2342:2004, Abschnitt 4.6.5.1.2, S. 7.

4.4 Beispielhafte Anwendungsfälle

An dieser Stelle werden einige beispielhafte Anwendungsfälle existierender Ontologien vorgestellt. Die exemplarische Auswahl wurde zum einen aufgrund der Erwähnung in der Literatur und zum anderen aufgrund ihres Beitrags zur Unterstützung des weiteren Argumentationsverlaufs in dieser Arbeit durchgeführt.[514] Erkenntnisse aus den Arbeiten zu diesen Ontologien fließen in die Entwicklung der FMEA-Ontologie ein. Gleichzeitig dienen sie als Beispiele für die in Kapitel 4.1.2 vorgenommene Gliederung nach Arten von Ontologien.

Die Ontologien Cyc und SENSUS dienen als Beispiele für Commonsense-Ontologien. Die KOWIEN-DMT-Ontologie dient als Beispiel für eine Domänen-Ontologie und die Frame-Ontologie dient als Beispiel für eine Repräsentations-Ontologie. Beispiele für die Arten Aufgaben- und Methoden-Ontologien werden aufgrund der Ähnlichkeit zu Domänen-Ontologien nicht vorgestellt. Weitere bekannte Ontologien sind zum Beispiel: Enterprise Ontology[515], WordNet[516], TOVE[517], Top-Level-Ontologie[518] und PhysSys[519].

4.4.1 Cyc-Ontologie

Im Bereich der Forschung zur Künstlichen Intelligenz wurde 1984 von Lenat ein Forschungsprojekt mit der Bezeichnung „Cyc“ ins Leben gerufen. Ziel des Projekts war es, Software-Applikationen bei unerwartet auftretenden – bei der Entwicklung (Programmierung) nicht vorhergesehenen – Abläufen durch den Aufbau einer umfassenden Wissensbasis ein Reagie-

514) Eine ähnliche Begründung für die Auswahl anhand einer Mischung aus „State-of-the-art“ und „unmittelbarem Nutzen für die Argumentation“ findet sich auch in Farrar, Bateman (2004), S. 1.

515) Vgl. Stader (1996); Uschold, Grüninger (1996); Uschold, King et al. (1998) und http://www.aiai.ed.ac.uk/project/enterprise/enterprise/ontology.html, Zugriff am 18.03.2007. Aus Sicht des Verfassers ist der Einsatz der Enterprise Ontology hier nicht möglich, weil bspw. die Bezeichnungen der Konzepte oft nicht ohne Erklärung nachvollziehbar sind (z. B. das Konzept „Qua-Entity“ als ein durch seine Relationen definiertes Konzept und „Misc-Spec-Detail“ als eine textuelle Erläuterung zu einem Konzept). Außerdem lässt sich anmerken, dass im engeren Sinne die Enterprise Ontology aufgrund der fehlenden Verknüpfungen der einzelnen „Sub-Ontologien“ keine zusammenhängende Konzeptualisierung einer Domäne darstellt.

516) WordNet ist ein lexikalisches Online-Referenzsystem, dessen Entwicklung durch aktuell angewendete psycholinguistische Theorien zur Funktionsweise des menschlichen lexikalischen Gedächtnisses inspiriert wurde. Es sind derzeit mehr als 140.000 Substantive, Verben, Adjektive und Adverbien der englischen Sprache in Synonymgruppen eingeordnet. WordNet wurde vom Cognitive Science Laboratory der Universität Princeton entwickelt (vgl. Miller, Beckwith et al. (1990), S. 235 ff., und http://www.cogsci.princeton.edu/~wn/, Zugriff am 18.03.2007).

517) Im TOVE-Projekt wurde die Bereitstellung einer einheitlichen Terminologie für Unternehmen durch eine Menge von Ontologien angestrebt. Unternehmenszusammenhänge sollten durch die formale Darstellung von Axiomen (Regeln) festgelegt werden. Eine Einführung in die TOVE-Ontologien findet sich in Fox, Grüninger (1997), S. 22 ff. In diesem Zusammenhang zu beachten ist auch das Vorgehensmodell des TOVE-Projekts in Kapitel 5.1.2.3.1.3, S. 139 f.

518) Die Top-Level-Ontologie von Sowa enthält sehr grundlegende Konzepte, die auf verschiedenen Ansätzen aus Logik, Philosphie und Künstlicher Intelligenz basieren. Vgl. Sowa (2000), S. 496 ff.

519) PhysSys wurde als „Ingenieur-Ontologie“ für die Modellierung, die Simulierung und das Design von physikalischen Systemen entwickelt. Vgl. Borst (1997), S. 367 ff.

ren auf diese unvorhergesehenen Abläufe zu ermöglichen.[520] Die Software-Applikationen sollten hierzu Zugriff auf Commonsense-Wissen erhalten.

Die Ergebnisse des Cyc-Projekts umfassen neben der Ontologie auch eine Wissensrepräsentationssprache (CycL).[521] Diese Sprache ermöglicht die formale Darstellung und damit Implementierung der Ontologie. Als weiteres Ergebnis entstand eine Inferenzmaschine, die automatisches Schlussfolgern über die Regeln und Fakten ermöglicht.[522] Bereits 1995 umfasste die Cyc-Ontologie über 100.000 Konzepte und eine Million Fakten und Regeln zu diesen Konzepten in ihrer Wissensbasis.[523] Im Jahr 2004 bestand die Cyc-Ontologie aus 200.000 Konzepten und zwei Millionen Fakten und Regeln zu diesen Konzepten.[524]

Die Cyc-Ontologie wird in Kategorien von Konzepten organisiert. Konzepte können auf unterschiedliche Arten spezialisiert werden, d. h. ein Unterkonzept kann mehreren Oberkonzepten zugeordnet werden. Auf der obersten Ebene wird bspw. das Konzept `Thing` unterteilt in die Konzepte `Individual` und `PartiallyIntangible`.[525]

4.4.2 Frame-Ontologie

Im Rahmen des Ontolingua-Projekts wurde am Knowledge System Laboratory (KSL) der Universität Stanford die Frame-Ontologie entwickelt, mit deren Hilfe Konzepte definiert und mit der Ontolingua-Software erkannt und übersetzt werden sollten.[526] Die gemeinsame Nutzung von Ontologien zu ermöglichen, die zwar ähnlich strukturiert sind, jedoch in verschiedenen Repräsentationssprachen vorliegen, war das Ziel dieser Ontologieentwicklung.[527] Damit wird die Frame-Ontology zu den Repräsentations-Ontologien gezählt. Sie stellt ein Rahmenwerk auf, mit dem die sprachlichen Ausdrucksmittel für die Formulierung einer (gemeinsamen) Ontologie definiert werden. Die Benennung der Konzepte und der Aufbau der Ontologie wurden eng an die Semantik der Sprache KIF angelehnt. Als Repräsentationssprache wurde jedoch Ontolingua verwendet.[528]

520) Vgl. Lenat, Guha et al. (1990), S. 32.

521) Siehe hierzu Kapitel 4.2.2.1.4, S. 106.

522) Vgl. hierzu http://www.cyc.com/cyc/technology/whatiscyc_dir/howdoescycreason, Zugriff am 18.03.2007.

523) Vgl. Lenat (1995), S. 33.

524) Vgl. Cycorp (2004).

525) Siehe hierzu und den weiteren Konzepten der oberen Ebenen der Cyc-Ontologie bspw. http://www.cyc.com/cycdoc/vocab/upperont-diagram.html, Zugriff am 18.03.2007.

526) Die Ontolingua-Software dient als Umgebung zur verteilten webbasierten Entwicklung von Ontologien. Neben einem Archiv bestehender Ontologien wird ein Editor zur Entwicklung und Bearbeitung von Ontologien bereitgestellt. Des Weiteren können Ontologien in die gleichnamige Sprache (siehe hierzu auch Kapitel 4.2.2.1.2, S. 105), aber auch in die Sprachen Prolog, LOOM und CLIPS übersetzt werden. Siehe hierzu Farquhar, Fikes et al. (1996), S. 44.3 ff. sowie im Internet unter http://www.ksl.stanford.edu/software/ontolingua/, Zugriff am 8.12.2006.

527) Siehe hierzu auch die Beschreibung der Frame-Ontologie in Gruber (1993), S. 212 ff.

528) Zu näheren Details und zum Verhältnis von KIF und Ontolingua siehe auch Kapitel 4.2.2.1.2, S. 105.

In Bezug auf Umfang und Mächtigkeit wird mit der Frame-Ontology von den Entwicklern nicht angestrebt (im Gegensatz etwa zu KIF[529]), die Strukturen aller existierenden Wissensrepräsentationsansätze abbilden zu können. Vielmehr dient die Frame-Ontology als konzeptuelle Basis für die Übersetzung von Ontologien innerhalb der Ontolingua-Software und kann daher als Spezifikation der ontologischen Verpflichtungen von Ontolingua angesehen werden. Vereinzelt werden in unregelmäßigen Zeitabständen noch immer Modifikationen an der Frame-Ontology vorgenommen.[530]

4.4.3 KOWIEN-DMT-Ontologie

Mit Hilfe des KOWIEN-Vorgehensmodells[531] wurde bei der Deutschen Montan Technologie GmbH (DMT)[532] in Essen mit Hilfe des Instituts für Produktion und Industrielles Informationsmanagement (PIM) der Universität Duisburg-Essen (Campus Essen) im Jahr 2004 eine Ontologie in F-Logic (und teilweise auch in DAML+OIL) entwickelt, die das Wissen über die Kompetenzen der DMT in Form von Konzepten, Relationen und Regeln abbildet. Die Ontologie hat zum Ziel, nicht nur den Mitarbeitern des Unternehmens ein gemeinsames Vokabular zur Beschreibung ihrer Kompetenzen bereitzustellen, sondern soll auch die computerverarbeitbare Explikation vorhandener Zusammenhänge und Einschränkungen in Bezug auf die vorhandenen Kompetenzen der DMT ermöglichen, die zum Teil zuvor nur implizit vorhanden waren.[533]

Von einem Einsatz der Ontologie im Rahmen eines Kompetenzmanagementsystems soll die DMT insofern profitieren, als zum Beispiel Mitarbeitern im Vertrieb und in der Personalabteilung eine detaillierte Übersicht über die Kompetenzen des Unternehmens und der Mitarbeiter bereitgestellt wird. Aufgrund der angestrebten Maßgaben *Wiederverwendbarkeit* und *Erweiterbarkeit* bei der Entwicklung der Ontologie könnte sie auch in anderen Unternehmen genutzt werden. Sie stellt mit ihrer Zielsetzung eine Domänen-Ontologie dar. Sie wird an dieser Stelle als Beispiel für eine Domänen-Ontologie aufgeführt, weil sie:

- eine erste Anwendung einer Ontologie im Bereich der betriebswirtschaftlichen Praxis darstellt,
- für ihre Domäne den Stand der Forschung widerspiegelt,
- in der Repräsentationssprache F-Logic vorliegt, die auch in dieser Arbeit Verwendung findet und

529) Siehe hierzu Fn. 528.

530) So datiert die letzte Änderung der Frame-Ontology vom September 2004.

531) Siehe hierzu Kapitel 5.4.4, S. 190 ff., und Apke, Dittmann (2003b); Apke, Dittmann (2004) sowie Apke, Dittmann (2005).

532) Aktuelle Informationen zum Unternehmen finden sich unter http://www.dmt.de, Zugriff am 18.03.2007.

533) Vgl. Apke, Bremer et al. (2004).

- mit dem KOWIEN-Vorgehensmodell entwickelt wurde und beide (Ontologie und Vorgehensmodell) wiederum maßgeblich vom Verfasser mitentwickelt wurden; Erkenntnisse dieser Arbeit basieren auf diesen Vorarbeiten.

In der Version 1.0 umfasst die Ontologie über 400 Konzepte, 130 Relationen und 40 Regeln. Hinzu kommen über 800 Instanzen. An ihrer tiefsten Stelle erreicht die taxonomische Darstellung der Konzepte die zehnte Ebene. Im Mittel erreicht die Ontologie die siebte Ebene in ihrer taxonomischen Darstellung.[534]

4.4.4 SENSUS

In den 90er Jahren wurde am Information Science Institute (ISI, Intelligent Systems Division) in Kalifornien eine linguistisch motivierte Ontologie entwickelt, die allgemeine Konzepte beinhaltet und deshalb zu den Commonsense-Ontologien gezählt wird.[535] Die Unterstützung Wissensbasierter Systeme für die maschinelle Übersetzung sollte das Haupteinsatzgebiet dieser groß angelegten Ontologie sein.[536] Als Entwicklungsgrundlage für SENSUS wurde eine Verschmelzung bereits existierender Ansätze angestrebt. Als Ansätze für die Entwicklung einer umfassenden, einheitlichen Ontologie wurden berücksichtigt: Penman Upper Model, ONTOS, Longman´s Dictionary of Contemporary English, WordNet und das Harper-Collins Spanish-English-Dictionary. Die SENSUS-Ontologie wurde in der Wissensrepräsentationssprache LOOM[537] entwickelt und umfasst 70.000 Konzepte.[538] Hiervon gehören etwa 400 Konzepte zur abstrakten oberen Ebene der Konzepthierarchie. Die als „Ontology Base“ bezeichnete obere Ebene entstand zunächst aus der Verschmelzung von PENMAN Upper Model mit ONTOS und beinhaltet die für die automatische Übersetzung notwendigen Generalisierungen. Etwa 50.000 weniger abstrakte Konzepte, die allgemeines Weltwissen berücksichtigen, werden der mittleren Ebene zugeordnet. Auf der unteren Ebene finden sich die restlichen Konzepte, die noch konkreter das allgemeine Wissen über die „Welt“ darstellen.

Die Entwickler von SENSUS haben einen Ansatz für die Entwicklung von „neuen“ Ontologien vorgestellt, der im Wesentlichen auf SENSUS als Ausgangsbasis fußt.[539] Hierzu dienen die Konzepte der unteren Ebene als Ausgangspunkt einer Ontologieentwicklung.[540]

Die SENSUS-Ontologie wurde seit ihrer Vorstellung nicht mehr weiter entwickelt und ist seither unter http://mozart.isi.edu:8003/sensus2/ zu erreichen (Zugriff am 8.12.2004).

---

534) Vgl. Apke, Bremer et al. (2004), Anhang B.

535) Siehe hierzu Kapitel 4.1.2, Seite 78 ff.

536) Vgl. Knight, Luk (1994), S. 773.

537) Ein Grund für die Verwendung von LOOM für die SENSUS-Ontologie dürfte darin gelegen haben, dass LOOM zuvor an demselben Institut (Intelligent Systems Devision, University of South California) entwickelt wurde. Siehe zu LOOM auch Kapitel 4.2.2.1.3, S. 105.

538) Vgl. Knight, Luk (1994), S. 773.

539) Siehe hierzu Kapitel 5.1.2.3.2.3, S. 152 f.

540) Aufgrund dieses Sachverhalts (Ausgangspunkt für eine Ontologieentwicklung) und weil sie darüber hinaus als Beispiel für eine Commonsense-Ontologie dient, wird die SENSUS-Ontologie vorgestellt.

5 Vorgehensmodelle

5.1 Vorstellung des Instruments Vorgehensmodelle

5.1.1 Grundlagen von Vorgehensmodellen

5.1.1.1 Allgemeine Grundlagen

Es lässt sich konstatieren, dass die Entwicklung von Software hauptsächlich als ein Prozess angesehen werden muss, der stark von der Kreativität menschlicher Entwickler abhängt.[541] Es werden hierzu Instrumente benötigt, die den Entwickler in diesem Prozess unterstützen, bspw. durch die Bereitstellung von Wissen über diesen Prozess.[542] Ein Vorgehensmodell kann als ein solches Instrument des Wissensmanagements angesehen werden. Vorgehensmodelle sind eine spezielle Form von Referenzmodellen,[543] denn Vorgehensmodelle enthalten Empfehlungen.[544]

Zusätzlich zu den Vorgehensschritten (Aktivitäten) soll ein umfassendes Vorgehensmodell auch eine Reihe von Methoden und Werkzeugen für jede Phase sowie eine Beschreibung der Beziehungen zwischen den Aktivitäten umfassen.[545]

Es werden die charakteristischen Merkmale ausgewählter Vorgehensmodelle erläutert und hinsichtlich ihrer Vor- und Nachteile diskutiert. Hierzu werden die generischen Vorgehensmodelle aus den drei Bereichen von Kapitel 2.1.2.5, S. 31 ff., analysiert. Die Analyse dient als Grundlage für das in Kapitel 6, S. 193 ff., vorgestellte OntoFMEA-Vorgehens-modell, das der Entwicklung von FMEA-Ontologien, bspw. für die Verwendung in einem Wissensbasierten System wie etwa OntoFMEA, Kapitel 7, S. 252 ff., dienen soll.

5.1.1.2 Grundlagen für Vorgehensmodelle des Software Engineering

Das Software Engineering befasst sich mit der Anwendung von ingenieurmäßigen Prinzipien bei der Erstellung von Software.[546] Durch den Einsatz von ingenieurmäßigen Prinzipien werden bei der Erstellung von Software die Ziele verfolgt, die vorgenannten[547] Nutzen stiftenden Punkte zu erreichen.

541) Vgl. Steinmann (2000), S. 19 ff.; Baskerville, Pries-Heje (1999), S. 177.

542) Vgl. Kneuper (2000), S. 108.

543) Vgl. zum Begriff „Referenzmodell“ Rosemann, Schütte (1999), S. 23 f.; Scheer (1999), S. 4 ff. und Schütte (1998), S. 69 ff.

544) Der Verfasser setzt hier voraus, dass Referenzmodelle einen normativen (empfehlenden) Charakter besitzen.

545) Vgl. zu diesen Ausführungen auch Uschold, King (1995), S. 2.

546) Vgl. Sommerville (2001), S. 6 f.; Bullinger, Fähnrich (1997), S. 6.

547) Siehe zu den Punkten der Nutzenstiftung S. 32.

Die Forschungsdisziplin des Software Engineering soll Methoden und Werkzeuge bereitstellen, die es erlauben,

1. einerseits *technische* Probleme
   (Spezifikation, Entwurf und Implementierung von Software) und

2. andererseits *organisatorische* Probleme
   (Projektorganisation und Schnittstellenmanagement[548]),

die bei der Erstellung von Software auftreten können, zu lösen und damit die intendierten Ziele zu erreichen.

Im Bereich des Software Engineering werden Vorgehensmodelle in vielfältigen Variationen beschrieben. Dabei wird in diversen Vorgehensmodellen ein idealisierter Ablauf des Softwareentwicklungsprozesses vorgestellt.[549] In diesen Vorgehensmodellen wird der Entwicklungsprozess in überschaubare Phasen zerlegt, wodurch die Planung (Analyse und Entwurf) und Realisierung (Durchführung und Kontrolle) eines Software-Projekts vereinfacht werden[550]. Die gesamten Phasen und die Ordnung ihrer zeitlichen Reihenfolge beschreibt man als Teil eines *Software-Life-Cycle*.[551]

In dieser Arbeit werden der sequentielle Software-Life-Cycle-Ansatz, der Wasserfall-Ansatz, der prototypbasierte Software-Life-Cycle-Ansatz und der Spiral-Ansatz vorgestellt.[552]

5.1.1.3 Grundlagen für Vorgehensmodelle des Knowledge Engineering

Alle Tätigkeiten und Überlegungen zur Erfassung, Verwaltung, Verwendung und Transformation von Wissen werden umfassend mit dem Begriff „Knowledge Engineering" belegt.[553]

548) Der Begriff „Schnittstellenmanagement" bezieht sich hier auf den ungehemmten vollständigen Informationsfluss zwischen bspw. funktional getrennten Organisationseinheiten eines Unternehmens.

549) Zum Tailoring, d. h. der Anpassung des Ablaufs eines Vorgehensmodells an spezifische Projektsituationen, vgl. Fischer, Biskup et al. (1998), S. 28 ff.

550) Es lässt sich an dieser Stelle anmerken, dass sich sämtliche Vorgehensmodelle in der Regel auf die vier groben Prozessphasen *Analyse, Entwurf*, *Realisierung* und *Einführung* zurückführen lassen oder auf Teilphasen hiervon.

551) Vgl. zum Begriff „Software-Life-Cycle" Kapitel 5.1.2.1.1, S. 124, und Pomberger, Blaschek (1993), S. 18 f.

552) Die genannten Ansätze finden weiteste Verbreitung in der Fachliteratur und wurden deshalb ausgewählt. Des Weiteren bauen die Ansätze des Knowledge und des Ontology Engineering auf diesen Ansätzen auf.

553) Puppe, Stoyan et al. (2003), S. 599. Diese „moderne" Definition geht in ihrer Bedeutung weit über die in der Vergangenheit aufgestellten Definitionen hinaus. So wurde von Waterman 1986 das *Knowledge Engineering* als der Prozess der Erstellung eines Expertensystems bezeichnet (vgl. Waterman (1986), S. 5). Dem folgten umfassendere Definitionen hinsichtlich des Software-Life-Cycles, wie z. B. die von Haun im Jahr 2000, der den gesamten Entwicklungs- und Wartungsprozess eines Expertensystems als *Knowledge Engineering* definiert (vgl. Haun (2000), S. 188). Vermutlich kann diese Erweiterung damit erklärt werden, dass Erfahrungen mit der Entwicklung von Wissensbasierten Systemen zeigen, dass Wissen in allen Phasen des *wissensbasierten* Software-Life-Cycles ingenieurmäßig erfasst, verwaltet, verwendet und transformiert werden muss.

Das Knowledge Engineering lässt sich bei Wissensbasierten Systemen hauptsächlich in eine Phase der *Wissensakquisition* und eine Phase der *Wissensverwendung* aufteilen.[554] Die Wissensakquisition kann wiederum hauptsächlich in die Unterphasen *Wissenserhebung, Wissensinterpretation* und *Wissensoperationalisierung* unterschieden werden.[555] Bei der Wissenserhebung wird das gemäß den Anforderungen relevante Wissen für den Einzelfall mittels Erhebungstechniken festgestellt.[556] In der Unterphase der Wissensinterpretation wird seitens der Entwickler das ermittelte Wissen hinsichtlich der grundlegenden Strukturen und möglicher abzubildender Schlussfolgerungsregeln untersucht. Die anschließende letzte Unterphase, die als Wissensoperationalisierung bezeichnet wird, umfasst die Implementierung der Ergebnisse der Wissensinterpretation als Wissen in einem lauffähigen Computersystem.

Die im Unterkapitel 5.1.2.1 vorgestellten und untersuchten „klassischen" Vorgehensmodelle des Software Engineering lassen sich grundsätzlich für die Neuentwicklung eines Wissensbasierten Systems anwenden. Sie werden jedoch nicht den spezifischen Anforderungen, die bei Wissensbasierten System auftreten, gerecht.[557] Insbesondere sind sie nur eingeschränkt geeignet für die Anforderungen, die sich aus der Wissensakquisition, d. h. beim Füllen des Systems mit „Leben", ergeben. Da innerhalb des Knowledge Engineering ein Schwerpunkt immer in der Wissensakquisition zu sehen ist, müssen speziellere Vorgehensmodelle zur Entwicklung herangezogen werden.[558] Dem Leser sei empfohlen, die Modelle aus Kapitel 5.1.2.1 zum Verständnis und als Basis für die Vorgehensmodelle des Knowledge Engineering zu berücksichtigen.[559]

Prinzipiell lassen sich zwei Ansätze des Knowledge Engineering unterscheiden. Zum einen werden der Prototyp-Ansatz und zum anderen der Modellbasierte Ansatz in der Literatur unterschieden.[560] Es werden diese beiden Ansätze des Knowledge Engineering vorgestellt.

5.1.1.4 Grundlagen für Vorgehensmodelle des Ontology Engineering

Ontology Engineering beschäftigt sich mit der prinzipiellen Entwicklung, Modifikation, Applikation und Evaluation von Ontologien.[561]

Es gab in den letzten Jahren zahlreiche Vorschläge[562] für Vorgehensmodelle zur Konstruktion von Ontologien, die jedoch teilweise von sehr unterschiedlichen Zielsetzungen und Voraussetzungen ausgingen.[563]

---

554) Vgl. Haun (200), S. 189.

555) Vgl. Sowa (2000), S. 452.

556) Eine Auswahl solcher Erhebungstechniken findet sich bspw. in Alan, Alparslan et al. (2003), insbesondere S. 18 ff.

557) Vgl. Angele, Fensel et al. (1998), S. 169 f.; Fernández López (1999), S. 4-1 ff.

558) Zu einer Übersicht an Vorgehensmodellen zur Entwicklung Wissensbasierter Systeme vgl. Angele, Fensel et al. (1998), S. 170 ff.

559) Zu einer ausführlichen Begründung siehe Haun (2000), S. 146 ff.

560) Vgl. bspw. Puppe, Stoyan et al. (2003), S. 602 ff., und Haun (2000), S. 196.

561) Vgl. Holsapple, Joshi (2002), S. 43.

Zum einen lassen sich vollständige Vorgehensmodelle zur Konstruktion von Ontologien heranziehen. Zum anderen finden sich (unvollständige) Ansätze für Vorgehensmodelle, die ihren Fokus lediglich auf einen Teilbereich der Ontologieentwicklung legen. Bspw. konzentrieren sich viele existierende Ansätze in der Literatur bei der Ontologieentwicklung auf die Verknüpfung mehrerer bereits bestehender Ontologien, so zum Beispiel SENSUS[564] und auch KACTUS[565]. Das bedeutet, sie haben ihren Fokus nur auf eine Teilproblematik innerhalb der Ontologieentwicklung gelegt.

An dieser Stelle werden diejenigen Ansätze vorgestellt, die für das zu entwickelnde Vorgehensmodell, somit insbesondere für die vollständige Entwicklung von Ontologien, von besonderer Relevanz sind. Aus der Untersuchung existierender Ansätze erhielt der Verfasser wertvolle Erkenntnisse als Grundlage zur Entwicklung eines Vorgehensmodells für die Konstruktion einer FMEA-Ontologie, die in einem ontologiebasierten FMEA-System in der betrieblichen Praxis eingesetzt werden soll. Die unvollständigen Ansätze, von denen der Verfasser weitere, jedoch nur punktuell verwertbare Informationen für die Entwicklung erhielt, werden kurz der Vollständigkeit halber im anschließenden Kapitel vorgestellt, jedoch nicht einer Bewertung unterzogen. Insgesamt liefern die Untersuchungen zu den Vorgehensmodellen Hinweise zu einer Entwicklung des OntoFMEA-Vorgehensmodells und damit zur Entwicklung der FMEA-Ontologie.

In den folgenden Kapiteln zu Vorgehensmodellen des Ontology Engineering werden sechs der wichtigsten Ansätze beschrieben, die wissenschaftlich anerkannt oder in der Praxis bewährt sind oder aufgrund ihrer Zielsetzung und der vorgeschlagenen Methoden besondere Bedeutung für die Konstruktion von FMEA-Ontologien haben.[566] Die Auswahl von Ansätzen wurde hinsichtlich ihrer Relevanz für das später zu entwickelnde Vorgehensmodell getroffen. Da die vorzustellenden Ansätze des Ontology Engineering oft sowohl zeitlich als auch inhaltlich aufeinander aufbauen, werden sie weitgehend in der chronologischen Reihenfolge ihrer Veröffentlichung (Jahresangabe rechts) dargestellt:[567]

---

562) In dieser Arbeit werden einige konkrete Beispiele näher erläutert (siehe hierzu das Ende dieses Kapitels). Hinweise zu ergänzenden Erläuterungen finden sich bspw. in Fn. 150, S. 33.

563) Die Notwendigkeit von Vorgehensmodellen zur Entwicklung von Ontologien ist in der Literatur unstrittig. Siehe bspw. Stuckenschmidt, van Harmelen (2005), S. 246.

564) Vgl. Swartout, Patil et al. (1996). Dieser Ansatz vereinigt u. a. die Ontologien *Penman Upper Model, ONTOS* sowie *WordNet*, um aus einer breiten, allgemeinen SENSUS-Ontologie domänenspezifische Ontologien abzuleiten. Siehe hierzu auch Kapitel 5.1.2.3.2.1, S. 149.

565) Vgl. zum Beispiel Schreiber, Wielinga et al. (1995). Die Autoren von KACTUS, einem Vorgehensmodell, das im Rahmen eines ESPRIT-Projekts entwickelt wurde, gehen von einer bereits bestehenden Wissensbasis aus, aus der durch Abstraktion generelle Ontologien, die schließlich zu einer „Gesamtontologie“ vereinigt werden, entwickelt werden. Siehe hierzu auch Kapitel 5.1.2.3.2.2, S. 150.

566) Damit geht der Verfasser über die meisten Untersuchungen zu diesem Themengebiet hinaus. So untersuchen bspw. Pinto, Martins (2004) lediglich drei Ansätze (TOVE-Ansatz, Enterprise-Ansatz und METHONTOLOGY-Ansatz) als die „most representative methodologies“ (Pinto, Martins (2004), S. 445). Siehe zusätzlich auch Fn. 727, S. 182.

567) Die beiden beschriebenen Vorgehensmodelle Enterprise-Model-Ansatz und TOVE-Ansatz wurden etwa zeitgleich veröffentlicht; hier wird jedoch der Enterprise-Model-Ansatz vorangestellt, weil er von diesen beiden den grundlegenderen Ansatz darstellt.

• IDEF5-Ansatz	1994
• Enterprise-Model-Ansatz	1995
• TOVE-Ansatz	1995
• METHONTOLOGY-Ansatz	1996
• On-To-Knowledge-Ansatz	2000
• Kollaborativer Ansatz	2002

5.1.2 Arten von Vorgehensmodellen

5.1.2.1 Vorgehensmodelle des Software Engineering

5.1.2.1.1 Der sequentielle Software-Life-Cycle-Ansatz

Mit dem Begriff „Software-Life-Cycle“ wird eine Zeitspanne von Beginn bis zum Ende der Benutzung einer Software verbunden. In Abbildung 22 ist das klassische sequentielle Phasenmodell[568], der Software-Life-Cycle, wiedergegeben.

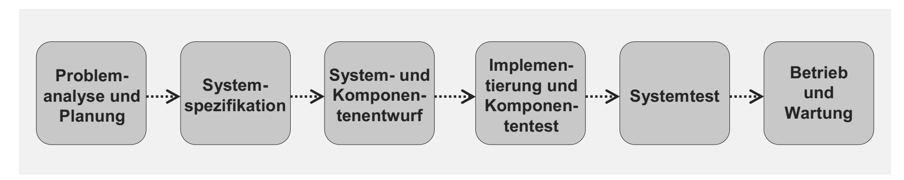


Abbildung 22: Phasenmodell Software-Life-Cycle-Ansatz[569]

Der sequentielle Software-Life-Cycle-Ansatz besteht aus den Phasen: *Problemanalyse und Planung*, *Systemspezifikation*, *System- und Komponentenentwurf*, *Implementierung und Komponententest, Systemtest, Betrieb* und *Wartung*. Der Grundgedanke des sequentiellen Software-Life-Cycle-Ansatzes ist, dass für jede der oben aufgeführten Phasen klar zu spezifizieren ist, welche Ergebnisse erwartet werden. Eine neue Phase wird erst dann begonnen, wenn

568) Vgl. dazu Pomberger, Blaschek (1993), S. 18ff.; Scacchi (2001), S. 5 ff. In dieser Arbeit wird der Begriff „Phasenmodell“ verwendet, wenn lediglich die (verkürzte) Darstellung der einzelnen Hauptschritte eines Vorgehensmodells angesprochen werden soll. Der Begriff „Vorgehensmodell“ bezieht sich hingegen immer auf den Gesamtkomplex der dahinter stehenden Methode, d.h. er umfasst bspw. auch Vorlagen.

569) In Anlehnung an Pomberger, Blaschek (1993), S. 18. Um eine Vergleichbarkeit der in diesem Kapitel vorgestellten Vorgehensmodelle zu gewährleisten, wird versucht, die Ansätze in eine einheitliche Modelldarstellung zu transferieren. Da der Fokus auf der Vergleichbarkeit liegt, wird auf eine genaue Transformation zu Gunsten der besseren Lesbarkeit verzichtet. Dies bringt mit sich, dass vorwiegend gestrichelte Kanten, die die Hauptflussrichtung von Informationen und die Reihenfolge der Phasen im zeitlichen Ablauf wiedergeben, verwendet werden, um den übergeordneten Bezug darzustellen. Des Weiteren werden einzelne Phasen (auch Hauptphasen, die mehrere Aktivitäten umfassen, und Einzelaktivitäten) durch abgerundete Kästchen dargestellt. Die Darstellung erfolgt gemäß den Quellen und soll jeweils lediglich der Orientierung dienen. Für eine genauere Darstellung (falls vorhanden) sei auf die betreffenden Quellen verwiesen, die jeweils angegeben werden.

die vorhergehende Phase abgeschlossen wurde, d. h. die spezifischen Ergebnisse erreicht wurden.

Im Folgenden werden die Ziele der einzelnen Phasen des sequentiellen Software-Life-Cycle-Ansatzes beschrieben:[570]

1. Problemanalyse und Planung
   Das Ziel der Problemanalyse und Planung ist es, den Problembereich der zu entwickelnden Software festzustellen und zu dokumentieren. Zur Problemanalyse und Planung gehören auch die Bestimmung der *finanziellen*, *personellen*, *technischen* sowie *zeitlichen* Ressourcen, die zur Realisierung der Software erforderlich sind.

2. Systemspezifikation
   In der Phase der Systemspezifikation wird zwischen dem Auftraggeber und dem Auftragnehmer (Softwareentwickler) festgelegt, was die zu entwickelnde Software leisten soll und welche Vorbedingungen für ihre Realisierung gelten. Die Anforderungen an die Software lassen sich in *funktionale* und *nicht-funktionale* Anforderungen unterteilen.[571]
   Die funktionalen Anforderungen definieren die vom Anwender erwarteten Systemfunktionen hinsichtlich des Einsatzzwecks der Software.
   Nicht-funktionale Anforderungen werden ebenfalls zumeist vom Benutzer an das System gestellt, berühren die eigentliche Funktionalität des Systems jedoch nicht. Diese Anforderungen werden zumeist implizit durch den Benutzer präsupponiert, wie z. B. eine hohe Zuverlässigkeit, größtmögliche Sicherheit und Portabilität sowie gewünschte Antwort- und Verarbeitungszeiten der Anwendung. Darüber hinaus enthalten die nicht-funktionalen Anforderungen die Spezifikation der Formen, in der die Anwender mit der Software kommunizieren (*Benutzerschnittstelle*). Als ein Kontrakt zwischen Auftraggeber und Auftragnehmer muss die Systemspezifikation für die Beteiligten *verständlich* formuliert sein. Als Grundlage für die Kommunikation zwischen Auftraggeber und Auftragnehmer, die aus unterschiedlichen Erfahrungskontexten stammen können (z. B. wirtschaftswissenschaftlich ausgebildeter Manager bzw. Informatiker), wurden Artefakte, wie bspw. semiformale Sprachen zur Modellierung, entwickelt.[572]

3. System- und Komponentenentwurf
   Das Ziel dieser Phase ist es festzulegen, welche Systemkomponenten entwickelt oder erworben werden müssen, um die in der Systemspezifikation identifizierten Anforderungen erfüllen zu können. Dabei stellt der Entwurf der Systemarchitektur

570) Vgl. für die einzelnen Phasen Pomberger, Blaschek (1993), S. 18-20. Die einzelnen Phasen werden als Gruppierungen von Aktivitäten im vorliegenden Fall verstanden.

571) Vgl. Pomberger, Blaschek (1993), S. 41 f.

572) Ein Beispiel für ein derartiges Artefakt ist die Unified Modeling Language (UML). UML stellt semiformale sprachliche Ausdrucksmittel bereit, mit denen der Auftraggeber und der Auftragnehmer die Anforderungen an die zu entwickelnde Software eindeutig beschreiben können. Vgl. zu UML Kapitel 5.2, S. 162 ff.

das hierbei wichtigste Ergebnis dieser Phase dar.[573] Die Systemkomponenten werden durch den Entwurf ihrer Schnittstellen, die Festlegung ihrer Interdependenzen, die Spezifikation ihres zugrunde liegenden logischen Datenmodells und den Entwurf ihrer algorithmischen Struktur spezifiziert.

4. Implementierung und Komponententest
   Das Ziel der Implementierung ist es, die im System- und Komponentenentwurf erhaltenen Ergebnisse formalsprachlich umzusetzen und damit auf dem Rechner ausführbar zu machen. Zunächst werden im Komponententest die einzelnen Komponenten hinsichtlich möglicher Fehler während des isolierten Ablaufs untersucht.

5. Systemtest
   Das Ziel des Systemtests ist es, die Interdependenzen der Softwarekomponenten unter realen Bedingungen zu prüfen, mögliche Fehler zu entdecken und zu beheben sowie zu gewährleisten, dass die Systemimplementierung die funktionalen und nicht-funktionalen Anforderungen erfüllt. Es wird sowohl verifiziert als auch validiert. Mit Hilfe der Verifikation wird der „korrekte" Ablauf der Anwendung überprüft und mit Hilfe der der Validation wird die „Korrektheit" der ausgegebenen Ergebnisse unter Berücksichtigung der Anforderungen, die an das System durch die Benutzer gestellt wurden, überprüft.[574]

6. Betrieb und Wartung
   Bei dieser Phase handelt es sich zeitlich gesehen um die längste Teilphase des Software-Life-Cycles. Nach Abschluss des Systemtests wird die Software für den Betrieb freigegeben. Die Aufgabe der Wartung ist es u. a., Fehler, die erst während des Betriebs der Software auftauchen, zu beheben sowie Erweiterungen der Software vorzunehmen.

5.1.2.1.2 Der Wasserfall-Ansatz

Der Wasserfall-Ansatz ist eine Modifikation des sequentiellen Software-Life-Cycle-Ansatzes.[575] Abbildung 23 zeigt das Phasenmodell des Wasserfall-Ansatzes mit einem Top-Down-Vorgehen.

Wie auch im sequentiellen Software-Life-Cycle-Ansatz werden hier die einzelnen Phasen als Gruppierungen von Aktivitäten verstanden, die jeweils vollständig bearbeitet und dokumentiert werden.

---

573) An dieser Stelle wird die Architektur eines Systems als das Zusammenspiel einzelner Komponenten einer Software verstanden.

574) Die Verifikation untersucht, ob das entwickelte System den nicht-funktionalen Anforderungen entspricht und die Validation untersucht, ob das entwickelte System den funktionalen Anforderungen der Benutzer entspricht. Hierzu werden oftmals die verkürzenden Merksätze „building the system right" und „building the right system" für die Verifikation bzw. Validation in der Literatur angeführt (vgl. bspw. Boehm (1989), S. 205, und van Vliet (2000), S. 49).

575) Vgl. zum Wasserfall-Ansatz Royce (1970).

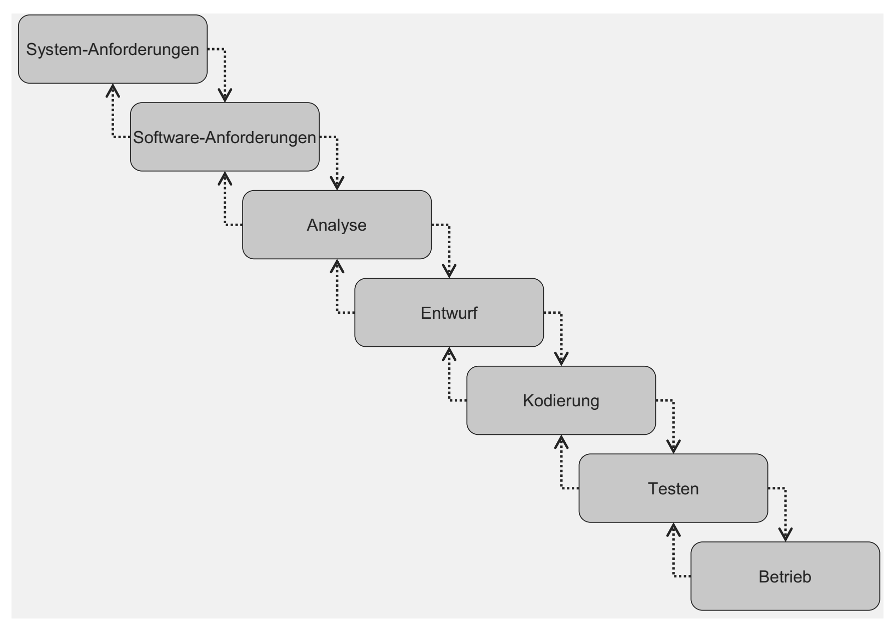


Abbildung 23: Phasenmodell Wasserfall-Ansatz[576]

Der Wasserfall-Ansatz enthält als eine Verfeinerung des sequentiellen Software-Life-Cycle-Ansatzes zwei grundlegende Modifikationen:[577]

1. Rückkopplung zwischen Phasen[578]
   Auf die strenge Sequentialisierung wird verzichtet. Stattdessen sind Rückkopplungen zwischen *aufeinander folgenden* Phasen vorgesehen.

2. Validierung
   Jede einzelne Phase (bspw. Entwurf oder Kodierung) wird mit einer Validierung abgeschlossen. Der Software-Life-Cycle wird somit um die (experimentelle) Validierung der Phasenergebnisse erweitert. Die Validierung, die innerhalb des sequentiellen Software-Life-Cycle-Ansatzes noch als Teilaspekt des Systemtests an-

576) Vgl. Biethahn, Muksch et al. (2000), S. 206. Bei dieser Darstellung finden sich die Phasen des vorangehenden Software-Life-Cycle-Ansatzes in etwa auf folgende Weise wieder: System- und Software-Anforderungen entsprechen der Problemanalyse und Planung; Analyse entspricht der Systemspezifikation; Entwurf entspricht dem System- und Komponentenentwurf; Kodierung entspricht der Implementierung; Testen entspricht dem Komponenten- und Systemtest; Betrieb entspricht Betrieb und Wartung.

577) Vgl. Pomberger, Blaschek (1993), S. 24.

578) Streng genommen findet sich die Rückkopplung erst in jüngeren Varianten des Wasserfall-Ansatzes. In den ursprünglichen Versionen wurde eine Rückkopplung nicht vorgesehen (vgl. Böhm, Wenger (1996), S. 10). In diesem Sinne wäre es präziser, beim Wasserfall-Ansatz von einer *Klasse* von Vorgehensmodellen auszugehen; für den weiteren Argumentationsverlauf wird diese Unterscheidung außer Acht gelassen, weil sie nur von untergeordneter Relevanz ist.

gesiedelt wurde, wird hier in die einzelnen Phasen integriert und so zu einem kontinuierlichen Prozessmerkmal erweitert.

Mit diesen beiden Modifikationen wird die streng sequentielle Vorgehensweise des klassischen Software-Life-Cycle-Ansatzes aufgehoben. Die Rückkopplungen insbesondere für die Systemspezifikation und den System- und Komponentenentwurf sowie die phasenweise Validierung sollen es ermöglichen, den Softwareentwicklungsprozess kontrolliert zu gestalten.

5.1.2.1.3 Der prototypbasierte Software-Life-Cycle-Ansatz

Im Gegensatz zu den bisher vorgestellten Ansätzen unterscheidet sich das prototypbasierte Vorgehensmodell durch die Überlappung bestimmter Phasen. Die Phaseneinteilung bleibt zwar, wie in Kapitel 5.1.2.1.1, S. 124, beschrieben, erhalten, jedoch mit dem Unterschied, dass die Phasen der Problemanalyse und Planung einerseits sowie der Systemspezifikation andererseits zeitlich überlappt ablaufen und die restlichen Phasen bis auf Betrieb und Wartung zu einer Phase integriert werden.[579] Die Phasen sind somit nicht Teilabschnitte einer diskreten Softwareentwicklung.[580]

Die Erstellung einer vereinfachten Version der Software oder Softwarekomponente (Prototyp) ist ein iterativer Prozess[581]: Dabei werden die beiden Phasen Prototyp-Spezifikation und -Konstruktion/-Test so lange wiederholt, bis der Anwender den Prototyp akzeptiert. Anhand dieses Prototyps wird mittels realen Einsatzbedingungen entsprechenden Experimenten untersucht, ob die eingangs aufgestellten Anforderungen erfüllt werden.

Der Vorteil dieser Vorgehensweise liegt nahe: Schon frühzeitig kann der Anwender ausprobieren, ob der Prototyp die Anforderungen an die Software erfüllt; bei Bedarf können frühzeitig Änderungen vorgenommen werden. Das Risiko einer fehlerhaften und nicht vollständigen Spezifikation der Anforderungen wird reduziert.

Es wird ein wesentlicher Unterschied zwischen klassischem Vorgehen und dem prototypbasierten Vorgehen deutlich: Während beim Erstgenannten sehr spät – nachdem alle Spezifikations- und Entwurfsprozesse abgeschlossen sind – implementiert wird, wird dagegen beim

579) Vgl. Pomberger, Blaschek (1993), S. 25; Wang (2002), S. 282. Dies ergibt eine zweiteilige Phasenaufteilung für den Prototyp-Ansatz: Prototyp-Spezifikation (Phasen Problemanalyse und Planung sowie Systemspezifikation) und Prototyp-Konstruktion/-Test (Phasen System- und Komponentenentwurf, Implementierung und Komponententest sowie Systemtest).

580) Bei einer diskreten Softwareentwicklung – wie etwa dem sequentiellen Software-Life-Cycle-Ansatz – wird ein zeitlich getrenntes Durchlaufen der einzelnen Phasen zur Bedingung.

581) Im Allgemeinen werden beim prototypbasierten Software-Life-Cycle-Ansatz (1) *exploratives* Prototyping, (2) *experimentelles* Prototyping und (3) *evolutionäres* Prototyping unterschieden. Das explorative Prototyping umfasst möglichst die vollständige Systemspezifikation anhand des Ziels (dem Einsatzzweck der Software). Hingegen wird beim experimentellen Prototyping eine vollständige Spezifikation von Teilsystemen (und damit Teilzielen) als Grundlage für die Implementierung angestrebt (vgl. dazu Sommerville (2001), S. 171 ff.). Beim evolutionären Prototyping wird eine initiale Implementierung vom Benutzer getestet und anschließend von den Entwicklern weiter verbessert. Es handelt sich somit um eine inkrementelle Systementwicklung (vgl. Pomberger, Blaschek (1993), S. 4).

Letztgenannten sehr früh – oftmals noch vor der endgültigen Festlegung aller Anforderungen – implementiert.

5.1.2.1.4 Der Spiral-Ansatz

Im Spiral-Ansatz[582] werden die bislang vorgestellten Ansätze integriert (bspw. wird die Entwicklung eines Prototyps durch entsprechend modellierte Prozessschritte unterstützt). Hierdurch wird die Möglichkeit geboten, eine den spezifischen Umständen eines Projekts zur Entwicklung einer Software gemäß den Anforderungen eines Auftraggebers entsprechende „spezifischere" Vorgehensweise auszuwählen. Die folgende Abbildung 24 zeigt diese Variante des Software-Life-Cycle-Ansatzes.

Während die Ausdehnung der Spirale in Abbildung 24 den bis zu einem bestimmten Zeitpunkt entstandenen Gesamtaufwand wiedergibt, beziehen sich die Winkeldimensionen der Spirale auf den Prozessschritt in den jeweiligen Spiralzyklen.

Der Spiral-Ansatz fasst den Entwicklungsprozess als iterativen Prozess auf, wobei jeder Zyklus folgende Aktivitäten enthält:[583]

1. Schritt planen
   Die Planung des ersten (bei Projektstart) oder nächsten Schritts (bei einem zuvor beendeten Zyklus) der Softwareentwicklung wird vollzogen.

2. Ziele, Alternativen und Randbedingungen festlegen
   In diesem Schritt erfolgt die Festlegung von Zielen, Alternativen und Randbedingungen als Anforderungen an das (Teil-)Produkt (im Sinne einer Anforderungsspezifizierung).

3. Alternativen bewerten, Risiken identifizieren und auflösen
   Die Alternativen zur Realisierung des (Teil-)Produkts werden bewertet. Dabei werden Risiken und Restriktionen (Kosten- und Zeitrestriktionen sowie Interdependenzen mit anderen organisatorischen Einheiten) identifiziert. Im Anschluss erfolgt die Evaluation der Alternativen hinsichtlich der Projektziele und unter dem Aspekt der identifizierten potenziellen Risikoquellen. Werden Risikoquellen identifiziert, schließen sich Maßnahmen zur Reduzierung der Risiken an. Am Ende dieses Schritts liegt ein Prototyp vor.

4. (Teil-)Produkt der nächsten Stufe entwickeln und evaluieren
   Das Produkt (Prototyp) der ersten oder nächsten Stufe wird entwickelt und verifiziert sowie gegebenenfalls auch validiert.

---

582) Der Spiral-Ansatz geht auf Boehm zurück; vgl. Boehm (1988) und Sommerville (2001), S. 53 ff.

583) Vgl. Sommerville (2001), S. 53 f.

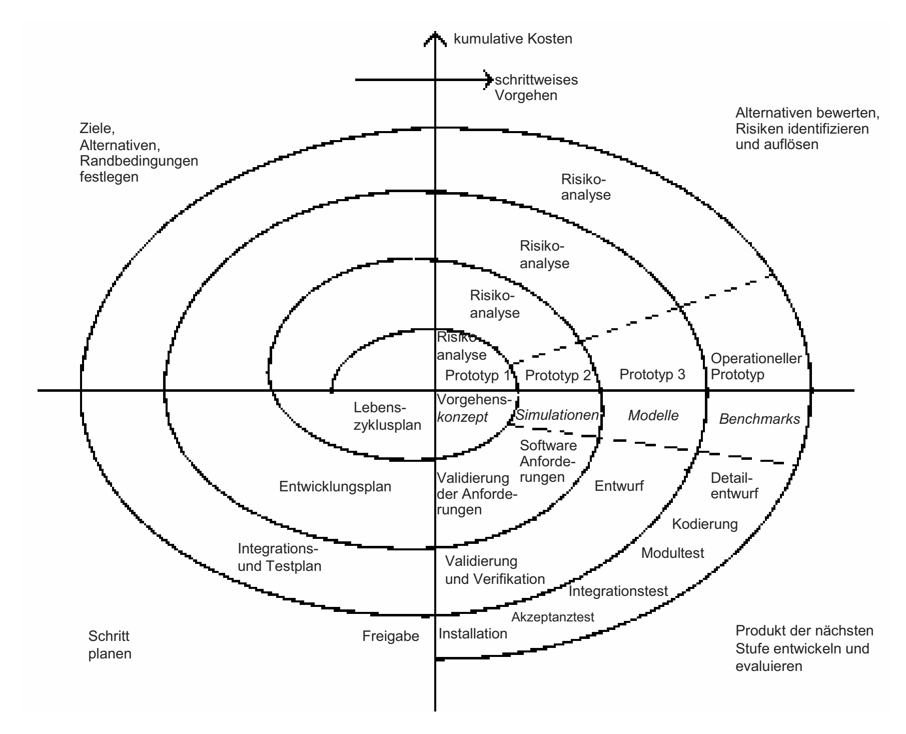


Abbildung 24: Phasenmodell Spiral-Ansatz[584]

Dieses Vorgehen wird im Uhrzeigersinn durchgeführt. Der Ablauf entspricht ungefähr dem klassischen Phasenmodell mit zusätzlichen Aktivitäten je Spirale: Risikoanalyse, prototypbasierte Evaluierung (Validierung der Anforderungen, Validierung und Verifikation des zweiten Prototyps, Modultest, Integrationstest, Akzeptanztest) und Festlegung des Projektplans für die nachfolgende Spirale (Lebenszyklusplan, Entwicklungsplan, Integrations- und Testplan). Insgesamt werden vier Spiralen durchlaufen:[585]

1. Konzept-Spirale
   Als die innerste Spirale wird das Vorgehenskonzept für die Softwareentwicklung detailliert festgelegt.

2. Simulationen-Spirale
   In dieser Spirale werden die Software-Anforderungen festgelegt und validiert. Hierzu wird die Funktionsfähigkeit der Software simuliert.

3. Modelle-Spirale
   Hier wird die Software als Modell entworfen und evaluiert.

584) Vgl. Biethahn (2000), S. 210.

585) Vgl. Myerson (1996), S. 201 ff. Der Vereinfachung halber wird an dieser Stelle von Spiralen gesprochen. Präziser wäre es jedoch, von einzelnen Zyklen einer Spirale zu schreiben.

4. Benchmarks-Spirale
   In der letzten Spirale wird die Software nach Erstellung eines detaillierten Entwurfs kodiert und ausgiebig auch mittels Benchmarks getestet.

Das Phasenmodell endet mit der Freigabe, wenn der inkrementelle Entwicklungsprozess eine als fertig angesehene Software mit deren Installation ergibt. Die Ausdehnung der Spiralen geben grob den kumulativen Kostenverlauf des Entwicklungsprozesses wieder.

5.1.2.2 Vorgehensmodelle des Knowledge Engineering

5.1.2.2.1 Prototyp-Ansatz

Bei diesem Ansatz wird frühzeitig ein Prototyp aus erkennbarem zu repräsentierendem Wissen als erste Implementierung eines Wissensbasierten Systems inklusive einer Wissensbasis und für die weitere Wissensakquisition entwickelt. Dies entspricht einem einfachen Transfermodell, bei dem ein mentales konzeptuelles Modell direkt (ohne die zwischenzeitliche Erstellung und Verwendung eines expliziten Modells) implementiert wird.[586]

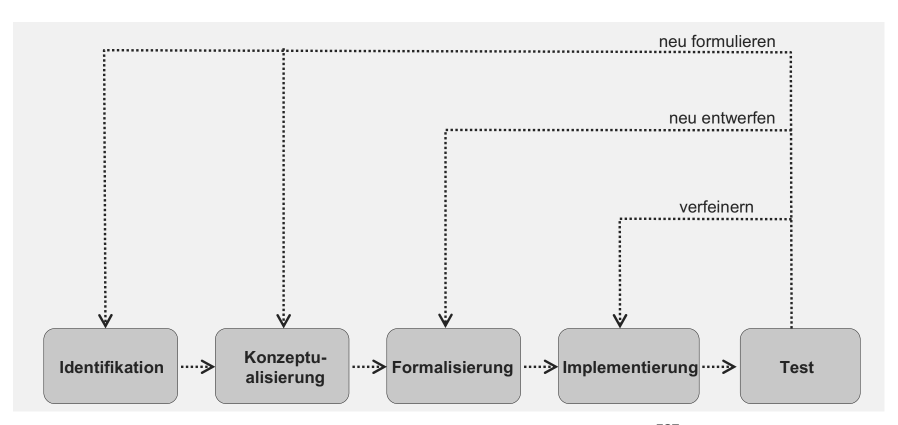


Abbildung 25: Phasenmodell Prototyp-Ansatz[587]

586) Deutlich wird hierbei der Zusammenhang zwischen dem Prototyp-Ansatz des Knowledge Engineering und dem Prototyp-Ansatz des Software Engineering: Beide versuchen möglichst früh ein lauffähiges Software-System zur weiteren Entwicklung zu erzeugen. Puppe, Stoyan und Studer bezeichnen ein solches Transfermodell als „naiv“ (vgl. Puppe, Stoyan et al. (2003), S. 603).

587) Vgl. Waterman (1986), S. 136.

Ein typisches Life-Cycle-Modell für den Prototyp-Ansatz des Knowledge Engineering beinhaltet in der Regel folgende Phasen (siehe auch Abbildung 25):[588]

1. Identifikation
   Die Anforderungen der späteren Benutzer und Entwickler werden identifiziert und dokumentiert.

2. Konzeptualisierung
   Es wird ein mentales konzeptuelles Modell für das spätere Wissensbasierte System erarbeitet. Die zu verwendenden Konzepte werden eruiert und definiert oder aus der Fachterminologie übernommen. Vor allem auf das Wissen der Domänenxperten wird zurückgegriffen. Schon in dieser oftmals sehr zeitaufwendigen Phase werden möglichst umfassend die Anforderungen an den Prototyp berücksichtigt.

3. Formalisierung
   Unter Berücksichtigung der entwickelten Konzeptualisierung wird eine formale Spezifizierung des bei der Konzeptualisierung identifizierten Wissens durchgeführt. Das informell konzeptualisierte Wissen wird formalsprachlich spezifiziert, um im Wissensbasierten System verarbeitet werden zu können. Hierzu müssen zusätzliche Anforderungen an die Wissensrepräsentation, die sich teilweise aus der Konzeptualisierung vererben können, berücksichtigt werden.

4. Implementierung
   An die Formalisierung des Wissens schließt sich die Erstellung des Prototyps an. Die formalsprachliche Spezifikation des Wissens wird in Form eines Prototyps in einer Software-Umgebung implementiert. Auf die erste Implementierung folgt die Verfeinerung der Implementierung gemäß den aufgestellten Anforderungen (siehe Punkt 5 – Test).

5. Test
   Mittels Verifikation und Validation wird getestet. Wenn im Testbetrieb entsprechend den aufgestellten Anforderungen Abweichungen festgestellt werden, wird der Prototyp gegebenenfalls verändert. Streng genommen durchläuft die Entwicklung somit einen schleifenförmigen (iterativen) Prozess der Schritte 4 und 5. Bei größeren Abweichungen von den Anforderungen kommt es zu einer neuen Formalisierung oder auch zur Neuformulierung von Inhalten der Identifikations- und der Konzeptualisierungsphase.

Bei dem beschriebenen Vorgehen werden die Beziehungen zum Software Engineering deutlich. Das Vorgehen weist größte Ähnlichkeiten zum prototypbasierten Software-Life-Cycle-

588) Vgl. Haun (2000), S. 200 ff. Dieser Ansatz stammt ursprünglich von Waterman (1986), S. 136 ff. Die Vor- und Nachteile des Prototypings wurden bereits in ähnlicher Form im vorhergehenden Unterkapitel zum Software Engineering erläutert. Um auf die spezifischen Unterschiede aufmerksam zu machen, wird an dieser Stelle nochmals darauf eingegangen, auch wenn sich der Verfasser hiermit dem Vorwurf der „Redundanz“ aussetzt; die eingeschätzte Wichtigkeit der Ausarbeitungen rechtfertigt in den Augen desselben jedoch diese Vorgehensweise.

Ansatz des Software Engineering auf. So werden bspw. die Phasen *Identifikation, Konzeptualisierung, Formalisierung, Implementierung* und *Test* so lange wiederholt, bis der Benutzer den Prototyp akzeptiert (siehe auch Kapitel 5.1.2.1.3, S. 128 f.).

Als Vorteile lassen sich für das beschriebene Vorgehen für den Prototyp-Ansatz erkennen:

- Wissensingenieur und Domänenexperte können den Wissenstransfer direkt miteinander abstimmen.[589]
- Der Domänenexperte kann das Verhalten des Systems überprüfen.
- Kürzere Feedback-Zyklen im Vergleich zum Software-Life-Cycle-Ansatz, bei dem der Domänenexperte das Verhalten des Systems erst gegen Ende der Entwicklung überprüfen kann, und rasche positive Ergebnisse erhöhen die Motivation der Beteiligten.

Als Nachteile lassen sich aufführen:

- Die Repräsentationsart des Wissens wird zu einem sehr frühen Zeitpunkt bestimmt.
- Es erfolgt keine standardisierte Phaseneinteilung des Kern-Entwurfsprozesses (Entwicklung des Wissensmodells). Die Wissensakquisition bleibt aufgrund des mentalen konzeptuellen Modells undeutlich.
- Eine sehr frühe Konzentration auf Implementierungsdetails bindet Entwicklungsressourcen, die bei der Konzeptualisierung und Anforderungserhebung oftmals effektiver eingesetzt werden könnten.
- Im Nachhinein ein mentales konzeptuelles Modell zu revidieren wird immer mit großen Schwierigkeiten verbunden sein, weil eine gemeinsame Diskussionsgrundlage (bspw. ein explizites Modell) fehlt.

5.1.2.2.2 Modellbasierter Ansatz

Historisch betrachtet bedeutet der Modellbasierte Ansatz des Knowledge Engineering eine Weiterentwicklung des Prototyp-Ansatzes des Knowledge Engineering.[590] Ein Wissensbasiertes System soll im Gegensatz zum Prototyp-Ansatz durch ein systematischeres Vorgehen erstellt werden, indem bspw. Prinzipien für die Modellierung vorgegeben werden. Der Modellbasierte Ansatz begreift die Wissensakquisition als einen Modellierungsprozess.

589) Der Wissensingenieur entwickelt das System, und das Wissen des Experten wird mittels Akquisitionstechniken im System hinterlegt.

590) Vgl. Puppe, Stoyan et al. (2003), S. 599 f.; Haun (2000), S. 204.

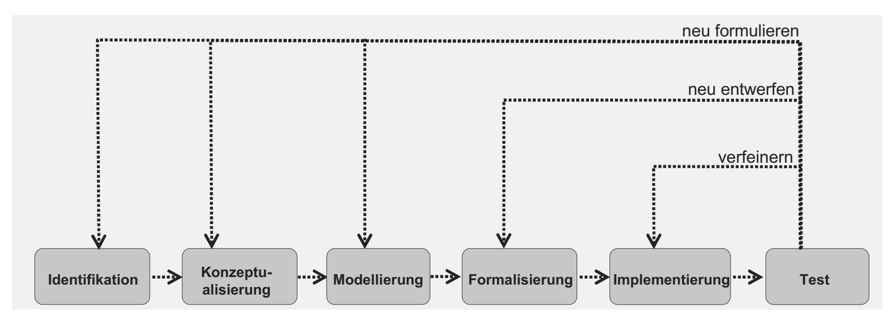


Abbildung 26: Phasenmodell Modellbasierter Ansatz

In der Praxis findet sich folgendes Life-Cycle-Modell (siehe Abbildung 26, S. 134):[591]

1. Identifikation
   Die Anforderungen der späteren Benutzer und der Entwickler werden identifiziert und dokumentiert.

2. Konzeptualisierung
   Die später im Modell verwendeten Konzepte werden eruiert und definiert oder aus der Fachterminologie übernommen. Vor allem auf das Wissen der Domänenexperten wird hierzu zurückgegriffen. Es existiert zunächst lediglich ein mentales konzeptuelles Modell. Schon in dieser oftmals sehr zeitaufwendigen Phase werden möglichst umfassend die Anforderungen an den Prototyp berücksichtigt.

3. Modellierung
   Das zu repräsentierende Wissen wird modelliert. Es erfolgt die Transformation des mentalen konzeptuellen Modells, d. h. das Wissen, das in der Praxis Anwendung durch Domänenexperten erfährt, wird für die Verwendung innerhalb des Wissensbasierten Systems explizit modelliert.

4. Formalisierung
   Das Wissen, das als explizites Modell vorliegt, wird formalsprachlich spezifiziert. Hierzu müssen zusätzliche Anforderungen an die Wissensrepräsentation, die sich teilweise aus der Konzeptualisierung und Modellierung vererben können, berücksichtigt werden.

5. Implementierung
   An die Formalisierung des Wissens schließt sich die Erstellung des Prototyps an. Als das Produkt des Formalisierungsprozesses wird die Spezifikation (als Ergebnis der Phase der Formalisierung) in einem Wissensbasierten System implementiert. Auf die erste Implementierung folgt die Verfeinerung der Implementierung gemäß den aufgestellten Anforderungen (siehe Punkt 6 – Test).

591) Der hier beschriebene Ablauf ergibt sich aufgrund der historischen Zusammenhänge aus der Erweiterung des Prototyp-Ansatzes (vgl. zur Phasenbeschreibung Haun (2000), S. 204 ff.).

6. Test
   Mittels Verifikation und Validation wird getestet, ob das System weiterentwickelt werden muss, um die aufgestellten Anforderungen zu erfüllen. Falls erforderlich, wird das Wissensbasierte System weiterentwickelt. Streng genommen durchläuft die Entwicklung somit – wie beim Prototyp-Ansatz – einen schleifenförmigen (iterativen) Prozess. Bei größeren Abweichungen von den Anforderungen kommt es zu einer neuen Formalisierung oder auch zur Neuformulierung von Inhalten der Identifikations- und der Konzeptualisierungsphase. Zusätzlich kann jedoch eine verbesserte Modellierung notwendig werden.

Der hauptsächliche Unterschied der Ansätze des Knowledge Engineering zum herkömmlichen Software Engineering liegt in einer in den einzelnen Punkten verfeinerten und spezialisierteren Arbeitsmethodik, die insbesondere Wissensanalyse (Wissenserhebung und Wissensinterpretation), generische Problemlösungsmodelle und heuristische Klassifikationen (in der Wissensoperationalisierung) umfasst.[592]

Als Vorteile des Modellbasierten Ansatzes lassen sich erkennen:

- Es erfolgt eine klare Trennung von Modellierung und Implementierung der Wissensbasis (hierdurch wird die Wiederverwendbarkeit der Wissensbasis für weitere Systeme deutlich verbessert, weil programmierspezifische Eigenheiten des Wissensbasierten Systems tendenziell weniger die Wissensbasis beeinflussen).

- Das explizite Modell ist nahe an der Terminologie des Domänenexperten, weil dieses Modell mit weniger Rücksicht auf die programmierspezifischen Eigenheiten des Wissensbasierten Systems entwickelt werden kann. Der Einfluss des Domänenexperten wird hierdurch tendenziell größer.

- Der Interpretationsprozess, d. h. die Überführung akquirierten Wissens in ein explizites Modell, wird transparent, d. h. für Dritte nachvollziehbar, gemacht.

Als Nachteile sind zu nennen:

- Modelle zur Implementierung in Wissensbasierte Systeme sind aufgrund ihrer Komplexität in der praktischen Anwendung nur von Fachleuten zu gebrauchen.

- Häufig fehlt eine ausreichende Werkzeugunterstützung, d. h., die praktische Anwendung wird durch fehlende Hilfsmittel erschwert.

- Modelle verfügen oft nur über eine gering ausgeprägte formale Semantik. Dies führt zu einer verzerrten Repräsentation des darzustellenden Wissens.

592) Vgl. Haun (2000), S. 205 f.

5.1.2.3 Vorgehensmodelle des Ontology Engineering

5.1.2.3.1 Vorgehensmodelle mit Gesamtfokus auf Ontologien

5.1.2.3.1.1 IDEF5-Ansatz

IDEF5 wurde 1994 von KBSI (Knowledge Based Systems, Inc.) entwickelt.[593] Der IDEF5-Ansatz wurde speziell entworfen, um die Entwicklung, die Modifizierung und die Instandhaltung von Ontologien zu unterstützen. IDEF steht hierbei als Akronym für **I**ntegrated Computer Aided Manufacturing **Def**inition und ist Teil einer Familie von Definitionen.[594]



Abbildung 27: Phasenmodell IDEF5-Ansatz

IDEF5 ist eine „Ontology (Description) Capture Method"[595], die einen strukturierten Ansatz zur Entwicklung von Domänen-Ontologien beinhaltet. Der Ansatz beinhaltet ein generelles Vorgehen mit folgenden Hauptphasen (siehe auch Abbildung 27):[596]

1. Organisation und Definition
   In einer ersten Phase wird die Organisation des Entwicklungsprojekts festgelegt. Des Weiteren werden projektspezifische Definitionen festgelegt. Hierzu gehören insbesondere auch die Festlegung des Zwecks, des Ausgangspunkts und der Zusammenhänge des Ontologieentwicklungsprojekts. Zusätzlich werden den Teammitgliedern Rollen zugewiesen.

2. Datensammlung
   Während der Datensammlung wird "rohes" Datenmaterial, das für die Ontologieentwicklung benötigt wird, akquiriert.[597]

3. Datenanalyse
   Die Datenanalyse wird betrieben zu dem Zweck, die Extraktion einer Ontologie aus den Daten zu erleichtern.

593) Vgl. KBSI (1994), S. 25 ff.

594) Siehe hierzu auch http://www.idef.com, Zugriff am 18.03.2007.

595) KBSI (1994), S. 1.

596) Vgl. http://www.idef.com/idef5.html, Zugriff am 18.03.2007.

597) Der Ansatz leidet hier, wie auch an anderen Stellen, an begrifflichen Ungenauigkeiten. Die Autoren gehen hier davon aus, dass sich aus Daten Wissen für eine Ontologie ableiten lässt. Weil an dieser Stelle die Darstellung des ursprünglichen Ansatzes im Sinne der Autoren im Vordergrund steht, wird dieser Umstand so belassen.

4. Initiale Ontologieentwicklung
   Die erste Ontologieentwicklung liefert eine präliminare Ontologie auf Basis der gesammelten Daten.

5. Validation und Verbesserung
   Die Ontologie wird validiert und gegebenenfalls verbessert, um den Entwicklungsprozess abzuschließen. Zum Zwecke der Validation wird die Ontologie instantiiert und das Ergebnis der Instantiierung wird mit der Struktur der entwickelten Ontologie abgeglichen.[598] Weist der Abgleich auf fehlende Übereinstimmungen hin, so ist die Ontologie an dieser Stelle zu verbessern.

Des Weiteren beinhaltet der IDEF5-Ansatz zwei Sprachen zur Repräsentation von Ontologien zur Unterstützung des Entwicklungsprozesses. Zum einen handelt es sich um eine *Schematic Language*, die speziell als grafische Sprache (vergleichbar etwa mit UML) entwickelt wurde, um übersichtlich und mit geringen Kenntnissen nachvollziehbar das Wissen einer Domäne abbilden zu können. Diese wird genutzt, um das Wissen der zu repräsentierenden Domäne zu modellieren. Der durchschnittliche Benutzer kann somit die erstellte Ontologie leicht nachvollziehen. Zum anderen gibt es eine Ausarbeitungssprache (*Elaboration Language)*. Sie stellt eine strukturierte textuelle Sprache dar, die es erlaubt, Elemente einer Ontologie detailliert zu charakterisieren.[599]

Der IDEF5-Ansatz enthält im zitierten Report-Anhang[600] eine strukturierte Bibliothek von üblicherweise benutzten Relationen, wie bspw. *zeitliche Relationen, Abhängigkeitsrelationen* und *Einflussrelationen*, auf die zurückgegriffen werden kann, um bspw. neue Relationen zu klassifizieren.

5.1.2.3.1.2 Enterprise-Model-Ansatz

Uschold und King[601] erkannten einen Mangel an strukturierten Vorgehensweisen bei der Ontologieentwicklung. Basierend auf den Erfahrungen, die sie bei der Entwicklung der „Enterprise Ontology" (Ontologie für Unternehmensmodellierungsprozesse[602]) gemacht haben, gehen Uschold und King auf einige grundsätzliche Aspekte ein, die bei der Konstruktion einer Ontologie beachtet werden sollen. So wird ein Grundgerüst für ein Vorgehensmodell vorgeschlagen, in dem die wichtigsten Schritte der Ontologieentwicklung aufgeführt sind und das als Ausgangspunkt für detaillierte Vorgehensmodelle dienen soll.

---

598) Vgl. KBSI (1994), S. 58.

599) Die *Elaboration Language* nutzt KIF als Basis, um eine Wiederverwendung und Integration bestehender Ansätze zu ermöglichen (vgl. KBSI (1994), S. 6). Das Akronym KIF steht für *Knowledge Interchange Format*, vgl. für eine Darstellung von KIF Genesereth, Fikes (1992). KIF wurde entwickelt, um den Austausch von Wissen zwischen verschiedenartigen Computerprogrammen zu erleichtern.

600) Siehe Fn. 595, S. 136.

601) Vgl. Uschold, King (1995), Uschold (1996a), Uschold (1996b).

602) Vgl. Uschold, Grüninger (1996), S. 56. Für die Entwicklung eines „Enterprise Models" werden unternehmensweite Daten-, Prozess- und Verhaltensmodelle zu einem konsistenten Modellsystem integriert, um eine Unternehmensrepräsentation zu bilden.

Ihr Ansatz definiert die wichtigsten Aktivitäten in Form einer Prozesskette. Nach der Bestimmung eines eindeutigen Einsatzzwecks werden die wichtigsten Konzepte gemeinsam vereinbart, in eine formale Sprache überführt und durch die Integration von bereits bestehenden Ontologien erweitert. Nach mehrfacher Revision und kritischer Beleuchtung (Evaluation) wird die Ontologie letztlich in ihre abschließende Form überführt. Während aller Aktivitäten wird eine ausführliche Dokumentation erstellt.

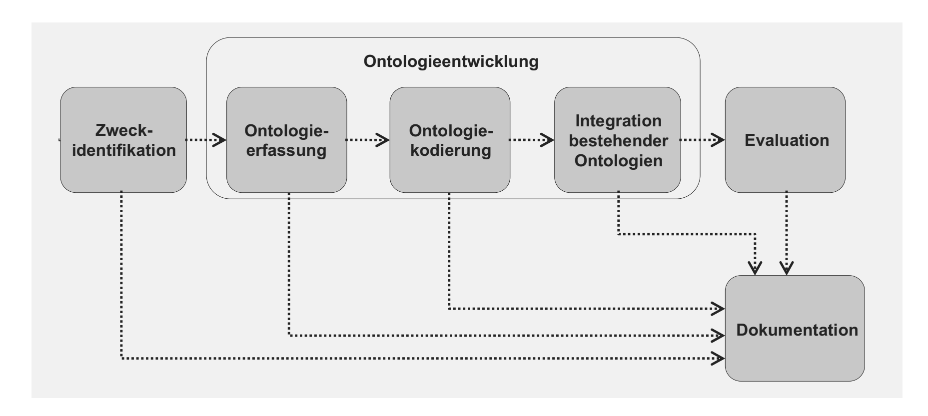


Abbildung 28: Phasenmodell Enterprise-Model-Ansatz

Die einzelnen Phasen des Vorgehensmodells werden im Folgenden kurz erläutert (siehe auch Abbildung 28).

1. Zweckidentifikation
   Zu Beginn des Entwicklungsprozesses soll zunächst festgelegt werden, zu welchem Zweck die Ontologie benötigt wird. Hierzu zählen Uschold und King auch die Festlegung, welche Anforderungen die Ontologie erfüllen soll. So besteht oftmals der oberste Zweck in der Wiederverwendung von Wissen. Aber auch die Art und Anzahl der Softwaresysteme, in denen die Ontologie eingesetzt werden soll, sind wichtige Anforderungen, die vor der Konstruktion der Ontologie zu entscheiden sind. Eine weitere Aufgabe, die von Uschold und King zur Zweckidentifikation und damit vor die eigentliche Entwicklung geordnet wird, ist die Identifizierung der Benutzer der Ontologie und ihrer Erwartungen.

2. Ontologieentwicklung
   Diese Phase wird weiter unterteilt in die Aktivitäten *Ontologieerfassung*, *Ontologiekodierung* und *Integration bestehender Ontologien*. Die Erfassung (Konzeptualisierung) beinhaltet die Identifikation der relevanten Konzepte und ihrer Beziehungen untereinander, die Darstellung der Konzepte durch präzise und eindeutige Textdefinitionen sowie die Festlegung von Bezeichnungen für diese Konzepte und ihre Beziehungen. Im Rahmen der Kodierung wird nach einer expliziten Repräsentation des konzeptualisierten Wissens durch eine formale Sprache gesucht. Dafür schlagen die Autoren zunächst die Definition einer *Meta-Ontologie* vor, die die Basiskonzepte für die Onto-

logiespezifikation enthält, um anschließend eine Repräsentationssprache auszuwählen und damit den Kode zu generieren. Im Anschluss an diese beiden vorangegangenen Aktivitäten stellt sich die Frage, ob und wie bereits existierende Ontologien integriert werden können. Für eine Integration bestehender Ontologien ist es wichtig, alle jeweils gemachten Annahmen für die jeweilige Ontologieentwicklung explizit formuliert zu haben. Hiermit lässt sich eine Übereinstimmung hinsichtlich der Integrationsmöglichkeiten zwischen den vormals verschiedenen Benutzergruppen erzielen.

3. Evaluation
   Für die Evaluation empfehlen die Autoren die Adaption von Methoden des Wissensmanagements.[603] Voraussetzung für die Bewertung einer Ontologie ist, dass zuvor ein Referenzrahmen entsprechend den Anforderungen aus der Zweckidentifikation festgelegt wurde, der als Grundlage für eine Evaluation der entwickelten Ontologie dient.

4. Dokumentation
   Um eine effektive gemeinsame Nutzung und Wiederverwendung einer Ontologie zu ermöglichen, sollen alle wichtigen Annahmen und Ergebnisse dokumentiert werden. Dazu zählen sowohl die Annahmen zu den Hauptkonzepten der Ontologie als auch die Annahmen für die Formulierung der Definitionen, also die Meta-Ontologie.

5.1.2.3.1.3 TOVE-Ansatz

Ziel des Projekts TOVE (Toronto Virtual Enterprise), das maßgeblich von Grüninger und Fox durchgeführt wurde, war die Entwicklung eines auf Ontologien basierenden Commonsense-Unternehmensmodells.[604]

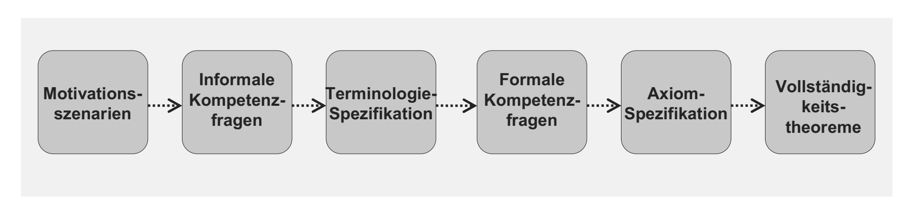


Abbildung 29: Phasenmodell TOVE-Ansatz

Aus den während des Projekts gewonnenen Erkenntnissen wurde anschließend ein umfassendes Vorgehensmodell formuliert, das Richtlinien für die Konstruktion von Ontologien sowie für die Evaluation ihrer Angemessenheit vorsieht. Kern des Vorgehensmodells ist die Durchführung der folgenden sechs Phasen:[605]

---

603) Die Phase der Evaluation bezieht sich gemäß Uschold und King vornehmlich auf die Validation einer Ontologie.

604) Vgl. Grüninger, Fox (1995), S. 1. Die Autoren berichten hierbei gar vom „Knowledge Engineering zweiter Generation“ im Hinblick auf das zu entwickelnde ontologiebasierte Unternehmensmodell. Für die Autoren gehören dabei Systeme, die lediglich die Regeln von Experten verarbeiten, zur ersten Generation des Knowledge Engineering.

605) Vgl. Grüninger, Fox (1995), S. 2 ff.

1. Motivationsszenarien
   Den Ausgangspunkt für die Ontologieentwicklung bilden die unterschiedlichen Probleme, die z. B. von Unternehmen identifiziert werden und durch eine Ontologie gelöst werden sollen. Sie werden in Form von Motivationsszenarien beschrieben, wobei zusätzlich auch erste intuitive Lösungsansätze zu präsentieren sind. Jeder Vorschlag für eine neue Ontologie oder die Erweiterung bestehender Ontologien soll mindestens ein solches Motivationsszenario beinhalten, um die Begründbarkeit anhand der Zwecke der Ontologie zu gewährleisten.

2. Informale Kompetenzfragen
   Aus den zuvor erfassten Szenarien wird eine Reihe von *Kompetenzfragen* abgeleitet. Die Kompetenzfragen sind für die Evaluation einer Ontologie von Bedeutung. Anforderungen, die als Wissen von der Ontologie erfüllt werden müssen, werden hier als Fragen ausformuliert. Die Autoren empfehlen eine hierarchische Strukturierung der informalen Kompetenzfragen und schlagen als Gliederung der ersten Stufe die Unterteilung in eine „Aktivitäten-Ontologie“ (Fragen zu den Aktivitäten in einem Unternehmen und zur Planung der Aktivitäten) und eine „Organisations-Ontologie“ (Fragen bezüglich möglicher organisationaler Einschränkungen dieser Aktivitäten) vor.

3. Terminologie-Spezifikation
   In dieser Phase wird die Terminologie der Ontologie definiert, indem für jede Kompetenzfrage alle zu ihrer Beantwortung notwendigen Konzepte, Attribute und Relationen identifiziert und spezifiziert werden. Dafür sollen die Beteiligten zunächst alle relevanten Konzepte festlegen und diese dann in einer formalen Sprache (etwa KIF) darstellen. Die Attribute der Konzepte werden anschließend in Form einstelliger Prädikate, die Relationen zwischen den Konzepten durch mehrstellige Prädikate repräsentiert.

4. Formale Kompetenzfragen
   Die zuvor nur informal formulierten Kompetenzfragen werden in dieser Phase in eine formale Darstellung, also in die für die Ontologiespezifikation gewählte Formalsprache, transformiert. Die Spezifizierung von formalen Kompetenzfragen ist eine Voraussetzung für die spätere Evaluation der Ontologie. Außerdem schränken sie die Menge der Axiome (siehe nächste Phase) ein, die später in die Ontologie aufzunehmen sind, indem sich aus ihnen ableiten lässt, welche Axiome notwendig gebraucht werden, um die formalen Kompetenzfragen beantworten zu können.

5. Axiom-Spezifikation
   Um die Definition der Begriffe der Ontologie und die Bedingungen ihrer Interpretation festzulegen, wird eine Menge von Axiomen definiert.[606] Sie spezifizieren die Semantik der Begriffe und werden formalsprachlich formuliert unter Anwendung der

606) Grüninger und Fox spezifizieren Axiome in einer Ontologie als die Einschränkung von Möglichkeiten der Interpretation zuvor definierter Konzepte (vgl. Grüninger, Fox (1995), S. 11). Der Verfasser bezeichnet in dieser Arbeit einen solchen Sachverhalt als non-deduktive Inferenzregel. Somit sind die genannten „Axiome“ den Inferenzregeln in dieser Arbeit gleichzustellen. Um die originäre Aussage der Autoren nicht zu sehr zu verwässern, wird in diesem Kapitel weiterhin von „Axiomen“ die Rede sein, auch wenn streng genommen Inferenzregeln gemeint sind.

Prädikate der Ontologie. Sie müssen notwendig und hinreichend sein für die Formulierung der formalen Kompetenzfragen und ihrer Antworten.[607] Wenn die vorgeschlagenen Axiome in dieser Hinsicht unzureichend sind, müssen in einem iterativen Prozess zusätzliche Axiome in die Ontologie aufgenommen werden oder auch nicht-notwendige Axiome eliminiert werden.

6. Vollständigkeitstheoreme
   Für die Evaluation der Ontologie werden zunächst die Bedingungen spezifiziert, unter denen die Antworten auf die Kompetenzfragen als vollständig zu gelten haben. Mit Hilfe dieser ebenfalls formalsprachlich formulierten Vollständigkeitstheoreme[608] können die Entwickler dann die Ontologie und die enthaltenen Axiome auf ihre Vollständigkeit in Bezug auf ihre Kompetenzfragen prüfen.

5.1.2.3.1.4 METHONTOLOGY-Ansatz

Der METHONTOLOGY-Ansatz, den Fernández, Gómez-Pérez und Juristo 1996 am Zentrum für Künstliche Intelligenz des Polytechnikums von Madrid erstmals veröffentlichten, soll eine umfassende Hilfestellung für die Konstruktion von Ontologien von Grund auf bieten („from the scratch“).[609] METHONTOLOGY orientiert sich stark an der IEEE-Norm 1074:1995.[610]

Von Gómez-Pérez[611] und Fernández et al.[612] wird der Ontologieentwicklung für die Zeit vor ihrem Ansatz eher das Prädikat eines Handwerks als das einer wissenschaftlichen Disziplin zugewiesen. Jedes Entwicklungsteam verwendet zuerst seine eigenen Entwurfskriterien und -phasen. Der dadurch entstehende Mangel an strukturierten, vereinheitlichten Vorgehensweisen beschränkt die Zusammenarbeit zwischen verschiedenen Entwicklungsteams. Daraus wird die Notwendigkeit eines „expliziten und vollständig dokumentierten Konzeptualisierungsmodells“[613] abgeleitet. Dazu präsentieren die Autoren eine Reihe von Aktivitäten, die Bestandteile des Ontologieentwicklungsprozesses sein sollen, sowie die Darstellung eines prototypbasierten Lebenszyklus für Ontologien. Teil dieses Rahmenwerks ist auch das METHONTOLOGY-Vorgehensmodell – eine detaillierte Beschreibung der Entwicklungsaktivitäten, wie sie durchzuführen sind, welche Techniken sich dafür eignen und welche Arte-

607) Zur Begründung führen Grüninger und Fox an: without the axioms we cannot express the question or its solution and with the axioms we can express the question and its solutions (Grüninger, Fox (1995), S. 11).

608) Abermals wäre es angebrachter, die englischsprachlichen Begriffe freier zu übersetzen. Der Begriff „Completeness Theorem“ würde eigentlich passender hier mit „Vollständigkeitssatz“ übersetzt werden, denn „Theorem“ ist in diesem Zusammenhang ein veralteter Begriff, der neben dem Englischen lediglich bei seit langem bekannten Sätzen im Deutschen noch gebräuchlich ist. Um jedoch nicht zu sehr von der originären Quelle abzuweichen, wird der Begriff „Vollständigkeitstheorem“ (nur) an dieser Stelle verwendet.

609) Vgl. für einen Überblick zum METHONTOLOGY-Ansatz Fernández, Gómez-Pérez et al. (1997), S. 33 ff.

610) Vgl. IEEE 1074:1995.

611) Vgl. Gómez-Pérez, Fernández et al. (1996), S. 41.

612) Vgl. Fernández López, Gómez-Pérez et al. (1999), S. 37; Fernández López (1999), S. 4.1 f.

613) Fernández López, Gómez-Pérez et al. (1999), S. 37.

fakte am Ende jeder Phase produziert werden. Dabei wird unterschieden zwischen den „technischen“ Aktivitäten der Ontologieentwicklung (die im oberen Teil der folgenden Abbildung aufgeführten Phasen *Anforderungsspezifizierung*, *Konzeptualisierung* und *Implementierung*) sowie den zusätzlich auszuführenden Unterstützungsaktivitäten (*Wissensakquisition*, *Integration, Evaluation* und *Dokumentation*).[614]

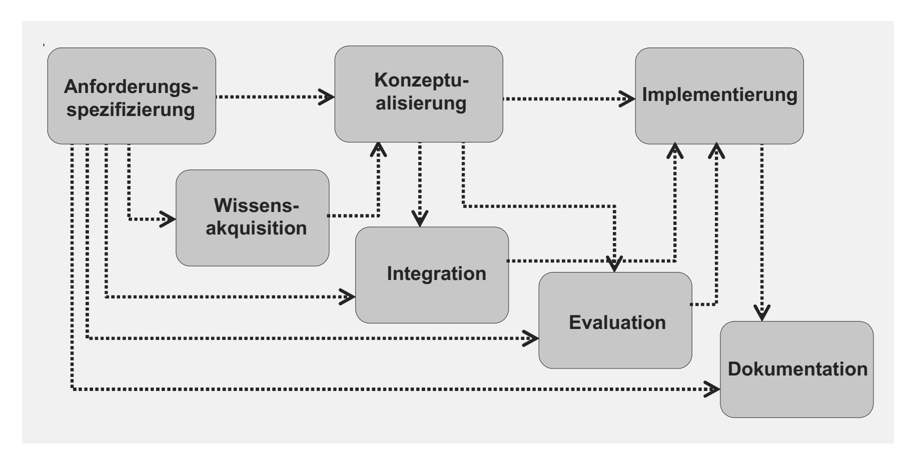


Abbildung 30: Phasenmodell METHONTOLOGY-Ansatz

Im Einzelnen werden die Phasen wie folgt durchlaufen:

1. Anforderungsspezifizierung
   Ziel dieser Phase ist die Erstellung eines natürlichsprachlich formulierten Ontologiespezifikationsdokuments, das die folgenden Informationen enthält: die intendierten Anwendungen und Benutzer der Ontologie, den Formalisierungsgrad, in dem die Ontologie implementiert werden soll, sowie den Umfang und die Granularität der Begriffsrepräsentationen. Mehrere Alternativen für das Vorgehen bei der Anforderungsspezifizierung sind möglich, z. B. Entwicklung von formalen Kompetenzfragen oder auch das Verfassen natürlichsprachlicher Texte mit einer Beschreibung der jeweiligen Anforderung.

2. Wissensakquisition
   Die Wissensakquisition bestehend aus Wissensidentifikation und -erfassung wird zumindest teilweise parallel zur Spezifizierung vorgenommen. In diesem Ansatz wird hierfür keine bestimmte Vorgehensweise vorgeschrieben. Stattdessen stellen die Autoren eine Reihe von möglichen Techniken vor, wie etwa Experteninterviews und Textanalysen. Anschließend sollen die relevanten Wissensquellen aufgelistet und die angewandten Erhebungstechniken beschrieben werden.

614) Vgl. Gómez-Pérez (2001), S. 9.

3. Konzeptualisierung
   Um das akquirierte Wissen durch ein konzeptuelles Modell strukturiert darstellen zu können, wird zunächst eine vollständige Termsammlung erstellt, die Konzepte und Verben[615] (einschließlich der Instanzen und Eigenschaften der Konzepte und Verben) der Domäne umfasst. Nachdem die akquirierten Terme in die beiden Kategorien *Konzepte* und *Verben* klassifiziert wurden, werden diese in Baumhierarchien weiter verfeinert, um weitere verwandte Konzepte oder Verben zu identifizieren und zu klassifizieren. Zur Darstellung empfehlen die Autoren an dieser Stelle informale Repräsentationsarten wie Verzeichnislisten und Tabellen.

4. Integration
   Da eines der wichtigsten Ziele der Ontologieentwicklung die Wiederverwendung von Wissen ist, sollen die Verantwortlichen bereits existierende Ontologien auf ihre Möglichkeiten zur Integration überprüfen. Durch die Untersuchung von Meta-Ontologien und die Analyse von Ontologien hinsichtlich ihrer Kohärenz zu der im eigenen Projekt erarbeiteten Konzeptualisierung kann beurteilt werden, ob und wie andere Ontologien integriert werden können. Als Resultat dieser Phase sieht METHONTOLOGY ein Integrationsdokument vor, das alle eventuell benutzten Meta-Ontologien, Ontologien und die daraus verwendeten Begriffe zusammenfasst.

5. Implementierung
   In diesem Schritt soll die informale Darstellung der Ontologie in eine formale Repräsentation transformiert werden. Die Wahl der Sprache ist dabei in diesem Ansatz offen. Es wird jedoch die Nutzung einer Entwicklungsumgebung für die Kodierung der Ontologie empfohlen, um die Fehlerwahrscheinlichkeit zu verringern und die Implementierung zu erleichtern. Am Ende der Phase liegt ein Formalisierungs- und Implementierungsdokument vor, das die Ergebnisse zusammenfasst.

6. Evaluation
   Vor ihrer Anwendung soll die Ontologie anhand eines Referenzrahmens – in diesem Falle die Ergebnisse der Anforderungsspezifizierung – untersucht und bewertet werden. Dazu zählt einerseits die Verifikation, also die Überprüfung der Korrektheit der Ontologie sowie ihrer Software-Umgebung und ihrer Dokumentation, andererseits die Validation, das heißt die Überprüfung der Eignung der Ontologie im Hinblick auf die gestellten Anforderungen der späteren Benutzer.

7. Dokumentation
   Während des gesamten Ontologieentwicklungsprozesses sollen die Beteiligten alle wichtigen Annahmen, Entscheidungen und Ergebnisse dokumentieren, damit die gemeinsame Nutzung und auch die Wiederverwendung der Ontologie erleichtert werden.

615) Fernández, Gómez-Pérez et al. unterscheiden abweichend zu dieser Arbeit Konzepte und Verben als Unterscheidungskriterien für Terme, die sämtliches „Wissen“ einer Domäne zusammenfassen. Verben repräsentieren dabei Aktionen innerhalb der betrachteten Domäne (vgl. Fernández, Gómez-Pérez et al. (1996), S. 38). Der Verfasser betrachtet hingegen auch Verben als Konzepte und vermeidet den Begriff „Term“ in Zusammenhang mit einer Konzeptualisierung (siehe hierzu auch Kapitel 2.1.2.4, S. 28 ff.).

Nach diesem Ansatz sollen die in jeder Phase entstehenden Schriftstücke (Anforderungsspezifikation, Wissensakquisitionsdokument, konzeptuelles Modell sowie Formalisierungs-, Integrations- und Implementierungsdokument) eine umfassende Dokumentation gewährleisten.

5.1.2.3.1.5 On-To-Knowledge-Ansatz

Der On-To-Knowledge-Ansatz ist Bestandteil eines Konzepts zur Entwicklung eines ontologiebasierten Wissensmanagementsystems, das erstmals im Jahr 2000 von Schnurr, Studer, Sure und Akkermanns vorgestellt wurde[616] und mittlerweile im deutschsprachigen Raum eine große Bekanntheit erlangt hat. In diesem Ansatz wird zwischen zwei Arten von Prozessen unterschieden: zwischen dem Prozess der Einführung und Instandhaltung von Wissensmanagementsystemen (*Wissens-Meta-Prozess*) und dem Prozess der Generierung, Erfassung und Nutzung des Wissens, der als *Wissensprozess* bezeichnet wird.[617] Der Wissens-Meta-Prozess umfasst gleichzeitig auch ein Vorgehensmodell zur Konstruktion von Ontologien und verfolgt dabei das Ziel, durch diese Einbettung der Ontologieentwicklung in die übergeordnete Wissensmanagementsystementwicklung die Anwendungsorientierung des Wissensmanagementsystems zu fördern.

Das im Folgenden dargestellte Vorgehensmodell beschreibt daher den gesamten Wissens-Meta-Prozess, der an die Vorgehensweise des „CommonKADS"-Ansatzes[618] angelehnt ist.

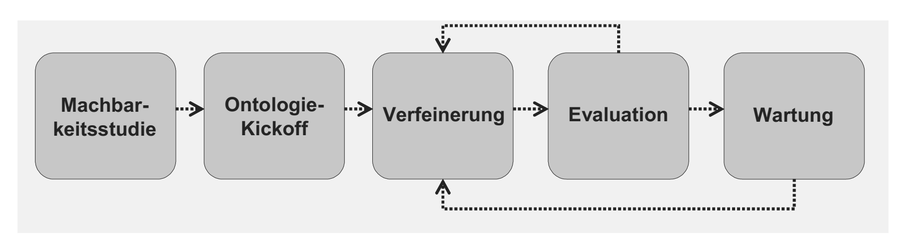


Abbildung 31: Phasenmodell On-To-Knowledge-Ansatz

Es finden sich die Phasen:[619]

1. Machbarkeitsstudie
   Bevor die Mitglieder des Projektteams mit dem „eigentlichen" Entwicklungsprozess beginnen, soll eine Machbarkeitsstudie durchgeführt werden, um die organisatorischen Rahmenbedingungen des zu entwickelnden Wissensmanagementsystems zu erfassen. Hauptbestandteil dieser Studie sind eine Marktanalyse sowie eine Analyse der unternehmensinternen Voraussetzungen. Mit Hilfe der Analysen sollen Chancen und Risiken aufgedeckt und dabei mögliche Entwicklungsalternativen skizziert werden. Durch

616) Vgl. Schnurr, Sure et al. (2000), S. 24 ff. Vgl. ferner auch Sure (2002) und Lau, Sure (2002), S. 124 ff.

617) Vgl. Sure, Staab et al. (2004), S. 118.

618) Vgl. zu CommonKADS Schreiber (2001) und Kapitel 5.4.3.1, S. 187 ff.

619) Vgl. Staab (2002), S. 203 ff., und Sure, Staab et al. (2004), S. 120 ff.

diese Informationen sollen Probleme, die einer Umsetzungsalternative im Wege stehen, und Möglichkeiten, die eine Umsetzungsalternative speziell ermöglicht, aufgezeigt und eine Entscheidung getroffen werden hinsichtlich der Eignung eines ontologiebasierten Wissensmanagementsystems für die Lösung der betrachteten Probleme. Die Machbarkeitsstudie dient als Grundlage für die Feststellung einer ökonomischen und technischen Machbarkeit des Projekts.

2. Ontologie-Kickoff
   Wenn die Machbarkeitsstudie mit einer Entscheidung für das Projekt abgeschlossen wurde, müssen im nächsten Schritt die Anforderungen an die Ontologie erhoben werden. Dazu wird mit Hilfe strukturierter Interviews eine Anforderungsspezifikation erstellt, die das Ziel der Ontologie, die Eingrenzung der Anwendungsdomäne, die potenziellen Benutzer und die Benutzeranforderungen als Katalog dokumentiert. Dieser Anforderungskatalog soll die Entwickler der Ontologie bei der hierarchischen Strukturierung der Konzepte unterstützen. Außerdem soll diese Phase eine Analyse der relevanten Wissensquellen (Domänenexperten, Organisationsdiagramme, Geschäftspläne, Wörterbücher, Indexlisten, Datenbank-Schemata etc.) umfassen sowie die Formulierung von Kompetenzfragen (als Fragen, die vom System beantwortet werden sollen). Darüber hinaus soll bereits eine erste informale (oder semi-formale) Beschreibung der Ontologie erstellt werden.

3. Verfeinerung
   In Zusammenarbeit mit Domänenexperten entwerfen die Ontologieentwickler eine semi-formale Basis-Ontologie, die dann schrittweise erweitert und verfeinert wird.[620] Dafür müssen alle relevanten Konzepte, Relationen zwischen den Konzepten und Regeln identifiziert und dargestellt werden und hinsichtlich ihrer Konsistenz und Vollständigkeit überprüft werden. Anschließend wählen die Entwickler eine Sprache zur formalen Repräsentation der Ontologie aus[621] und transformieren die Elemente der semi-formalen *Basis-Ontologie* in formale Konzepte, Relationen und Regeln der *Ziel-Ontologie*. Es lassen sich somit die folgenden Teilphasen unterscheiden:

   a. Erstellung einer ersten semi-formalen *Basis-Ontologie*, die alle relevanten Konzepte und Relationen aus der Kickoff-Phase enthält[622],

620 Zu Anfang der Ontologieentwicklung soll schnell eine erste Vorstellung von der Ontologie gewonnen werden, um die Anforderungserfüllung frühzeitig justieren zu können und die einzubeziehenden Wissensquellen festzulegen (vgl. Staab (2002), S. 203 f.). Dieses Vorgehen zieht in der Regel eine zumeist unvollständige Ontologie nach sich, die schließlich weiter verfeinert werden muss.

621) Die Autoren schlagen hierzu Sprachen wie RDF oder DAML+OIL vor. Aktuelle Informationen zu den Sprachen RDF (Resource Description Framework) und DAML+OIL (DARPA Agent Markup Language + Ontology Inference Layer) sind zu finden unter http://www.w3.org/RDF/ bzw. http://www.w3.org/TR/daml+oil-walkthru/ (Zugriffe am 18.03.2007) sowie in den Kapiteln 4.2.2.2.2 bzw. 4.2.2.2.3, S. 108.

622) Bei der Terminologie-Spezifikation nach Grüninger und Fox, S. 140, wird die Erstellung einer Taxonomie bei der Festlegung der Terme im Gegensatz zu der hier definierten Phase nicht vorrangig behandelt.

b. Akquisition von Wissen von Domänen-Experten mit dem Ziel der Verfeinerung der ersten semi-formalen *Basis-Ontologie*, die schließlich alle relevanten Konzepte, Relationen zwischen den Konzepten und darauf aufbauende Regeln enthält,

c. Formalisierung der Basis-Ontologie in eine erste *Ziel-Ontologie*, die formalsprachlich repräsentiert wird.

4. Evaluation
In der Phase „Evaluation" wird die entwickelte Ontologie in Hinblick auf ihren Nutzen in dem vorgesehenen Wissensmanagementsystem bewertet. Zunächst überprüfen die Entwickler, ob die *Ziel-Ontologie* alle Kriterien der Anforderungsspezifikation erfüllt und ob sie die festgelegten Kompetenzfragen „beantworten" kann. Darüber hinaus wird die Ontologie anhand eines Prototyps des Wissensmanagementsystems in ihrer späteren Anwendungsumgebung getestet, um durch Rückmeldungen seitens der testenden Benutzer Fehler zu identifizieren. Falls sich durch die Evaluation ein Revisionsbedarf ergibt, müssen die Fehler durch eine Rückkehr in die Verfeinerungsphase behoben werden.

5. Wartung
Schnurr, Sure et al.[623] empfehlen eine regelmäßige Anpassung der Ontologie, um Änderungen in der Umgebung Rechnung zu tragen. Da die Wartung von Ontologien nach Meinung der Autoren in erster Linie ein organisatorischer Prozess ist, sollten Regeln hierfür formuliert und die Verantwortlichkeiten zur Durchführung der Modifikationen festgelegt werden (bspw. um neue Versionsstände der Ontologie freizugeben). Für größere Änderungen wie z. B. Umstrukturierungen ist oft ein erneuter Durchlauf der Verfeinerungs- und der Evaluationsphase erforderlich. Analog zur ersten Verfeinerungsphase soll das Feedback von den Benutzern wichtige Hinweise für erforderliche Änderungen an der *Ziel-Ontologie* liefern. Nach Inbetriebnahme des ontologiebasierten Wissensmanagementsystems müssen nicht nur die Ontologien, sondern auch das darauf aufbauende System gewartet werden. Sowohl Modifikationen der Ontologie als auch Änderungen am Aufbau des Systems können hierbei im organisatorischen Kontext berücksichtigt werden.

5.1.2.3.1.6 Kollaborativer Ansatz

Holsapple und Joshi verfolgen in ihrem Ansatz zur Ontologieentwicklung das Ziel, den Aspekt der Bindung der unterschiedlichen Beteiligten an die Ontologie möglichst früh in den Entwicklungsprozess zu integrieren, um so die Akzeptanz und die gemeinsame Nutzung der

623) Vgl. Schnurr, Sure et al. (2000).

Ontologie zu fördern.[624] Die Autoren unterscheiden fünf Klassen verschiedener Ansätze mit unterschiedlichen Ausgangspunkten für das Design von Ontologien[625]:

- *Inspiration* (ausgehend von einer individuellen Sicht auf die Domäne),
- *Induktion* (mit einem konkreten Fall als Grundlage für eine Verallgemeinerung),
- *Deduktion* (basierend auf generellen Prinzipien, die angepasst werden),
- *Synthese* (die Zusammenfassung bestehender Ontologien zu einer umfassenderen Ontologie) und
- *Kollaboration* (Einbringung der Ansichten und Erfahrungen mehrerer Personen).[626]

Jede dieser Vorgehensweisen hat Vor- und Nachteile und ist jeweils für eine bestimmte Situation besser geeignet als eine andere.[627]

Holsapple und Joshi haben sich bei der Durchführung einer Fallstudie für den Kollaborativen Ansatz entschieden, weil mit der Beteiligung einer umfangreichen Anzahl von Testpersonen die Chancen für die spätere Akzeptanz der Ontologie steigen. Auch die Risiken einer lückenhaften Ontologiespezifikation können so verringert werden.[628]

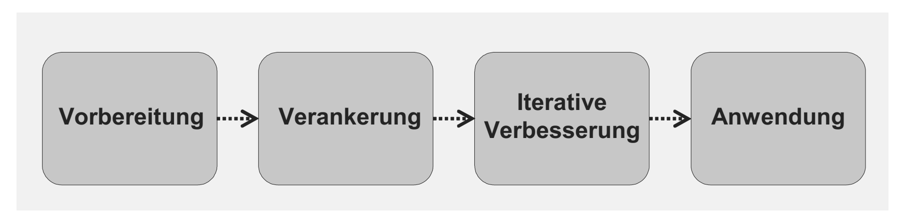


Abbildung 32: Phasenmodell Kollaborativer Ansatz

624) Vgl. Holsapple, Joshi (2002), S. 43.

625) Vgl. Holsapple, Joshi (2002), S. 44. Die fünf Klassen von Ansätzen können auch miteinander kombiniert werden.

626) Diese Klassen von Ansätzen können auch für die Klassifizierung der hier behandelten Vorgehensmodelle zur Ontologieentwicklung verwendet werden. So baut der TOVE-Ansatz bspw. auf der *Inspiration* auf. Das Konzept der *Kollaboration* ist in den meisten Ansätzen zu finden. Auf einer *Synthese* basierende Vorgehensmodelle werden in dieser Arbeit nicht betrachtet, weil die Neuentwicklung von Ontologien im Fokus der Untersuchung liegt.

627) Siehe zu einem kurzen Aufriss der Vor- und Nachteile Holsapple, Joshi (2002), S. 45.

628) Da sich außer zum *Kollaborativen Ansatz* zu den verschiedenen weiteren Klassen von Holsapple und Joshi bisher keine Anwendungen nachweisen lassen, wird im Folgenden lediglich der *Kollaborative Ansatz* näher betrachtet. Dieser Umstand liegt wahrscheinlich im zarten Alter der Erkenntnisse von Holsapple und Joshi begründet, die bei den hier betrachteten Ausführungen den jüngsten Ansatz darstellen. Das Phasenmodell *Kollaborativer Ansatz* kann insofern für alle genannten Ansatzmöglichkeiten von Holsapple und Joshi gelten.

Es lassen sich vier Phasen im Kollaborativen Ansatz ausmachen:[629]

1. Vorbereitung
   In dieser ersten Phase werden Entwicklungskriterien, wie z. B. Verständlichkeit, Korrektheit, Prägnanz, Klarheit und Nutzen, definiert[630], die einerseits als Richtlinien während der Entwicklung und andererseits für die Bewertung der Qualität einer Ontologie im Nachhinein notwendig sind. Anschließend werden Bedingungen festgelegt, um die Anwendung der Entwicklungskriterien auf bestimmte Bereiche einzuschränken. Die vorgestellte Fallstudie konzentriert sich auf betriebliches Wissensmanagement in Unternehmen und auf ein Top-Down-Vorgehen bei der Ontologieentwicklung. Außerdem werden in der Vorbereitungsphase Evaluationsstandards bestimmt. Genannt seien hier bspw.: existierende Ontologien, Wissensmanagementkonzepte und Forschungsergebnisse, zu denen der Entwicklungsprozess und die resultierende Ontologie konsistent sein müssen.[631]

2. Verankerung
   Jede der verbliebenen vier Klassen von Entwicklungsansätzen kann für die Formulierung einer Basis-Ontologie, die als Ausgangspunkt für die weitere kollaborative Ontologieentwicklung dient, angewendet werden. Die Basis-Ontologie stellt einen ersten „Anker" zur Orientierung auf dem Weg zu einer vollständigen Ontologieentwicklung dar. Die Autoren entscheiden sich bei der Entwicklung der Basis-Ontologie für den *Synthese*-Ansatz, bei dem sie die Konzepte und Relationen aus den Evaluationsstandards, die in der vorangegangenen Phase „Vorbereitung" festgelegt wurden, konsolidieren und neu zu einer Ontologie organisieren. Diese Basis-Ontologie wird in mehreren Iterationen als „Anker-Ontologie" durch einen Abgleich mit den Entwicklungskriterien gemeinsam mit allen Beteiligten (kollaborativ) weiterentwickelt (siehe auch *iterative Verbesserung*).

3. Iterative Verbesserung
   In dieser Phase wird die Delphi-Methode[632] eingesetzt, um die Basis-Ontologie schrittweise zu verbessern. Zunächst werden dafür bestimmte Teilnehmer (vor allem die Benutzer) identifiziert und in Ausschüssen gruppiert. Die Ontologieentwickler senden die Basis-Ontologie an die Mitglieder der Ausschüsse, damit diese den ersten Entwurf kommentieren. Nachdem die Bewertungen der Ontologie zurückgeschickt worden sind, findet eine Revision statt, um Kritik und Verbesserungsvorschläge in die

629) Vgl. Holsapple, Joshi (2002), S. 45.

630) Die genannten Kriterien wurden Holsapple, Joshi (2002), S. 45, entnommen (dort: comprehensiveness, correctness, conciseness, clarity und utility). Leider bleiben die Autoren eine Operationalisierung ihrer „Gütekriterien" schuldig, so dass deren Angemessenheit durch Dritte nicht nachvollzogen werden kann.

631) Vgl. Holsapple, Joshi (2002), S. 45. Wie schon in der voranstehenden Fn. bemängelt, bleiben Holsapple und Joshi eine Präzisierung und Nachvollziehbarkeit ihrer Ausführungen schuldig. So lässt sich für den Leser nicht erschließen, ob alle Evaluationsstandards gleich erstrebenswert sind oder weshalb diese überhaupt erstrebenswert sind.

632) Vgl. Linstone, Turoff (1975). Die Delphi-Methode ist eine strukturierte Vorgehensweise für die Befragung von Experten und die Sammlung und die Integration der verschiedenen Sichten auf eine bestimmte Problemstellung.

Ontologie einzuarbeiten. Anschließend wird die modifizierte Version erneut an die Teilnehmer gesendet. Dieser Prozess wird iterativ so lange durchgeführt, bis die Ontologie von allen Beteiligten übereinstimmend als vollständig und den Anforderungen (den Entwicklungskriterien) entsprechend angesehen wird. Um die verschiedenen Kommentare und Vorschläge zu verwalten und umzusetzen, teilen die Entwickler die Antworten in Kategorien ein (z. B.: *begründet und häufig, gelegentlich, selten und zufällig*) und fertigen ein Dokument zur Zusammenfassung der Kritik an. Die Modifikationen der nächsten Iterationsphase sollen sich hieran orientieren.

4. Anwendung
   Abschließend wird der Nutzen der Ontologie durch ihre Anwendung in konkreten Projekten überprüft. Die Autoren setzen ihre Ontologie z. B. als Rahmenwerk bei der Untersuchung von Wissensselektionsprozessen und Wissensselektionstechniken und bei der Generierung eines „Wissenskettenmodells" ein.[633]

5.1.2.3.2 Vorgehensmodelle mit spezifischem Teilfokus auf Ontologien

In diesem Kapitel werden Vorgehensmodelle berücksichtigt, die sich nur mit Teilen einer Ontologieentwicklung dezidiert auseinandergesetzt haben. Sie besitzen somit nicht den umfassenden Anspruch wie die vorher genannten Ansätze an die Entwicklung von ontologiebasierten Systemen über den gesamten Life-Cycle. Jedoch erscheinen einige Erkenntnisse als so wertvoll, dass sie an dieser Stelle kurz vorgestellt werden. Folgende Ansätze werden dabei wiederum in chronologischer Reihenfolge berücksichtigt:

• MENELAS-Ansatz	1994
• KACTUS-Ansatz	1995
• SENSUS-Ansatz	1996
• ONIONS-Ansatz	1996
• OntoCLEAN-Ansatz	2002

5.1.2.3.2.1 MENELAS-Ansatz

Die MENELAS-Ontologie wurde von Bouaud, Bachimont et al.[634] vorgestellt. Während ihrer Entwicklung wurden insbesondere vier Prinzipien ermittelt, die wichtig für die taxonomische Abbildung von Wissen innerhalb von Ontologien sind. Das Ziel von MENELAS war die Entwicklung eines Wissensbasierten Systems, das auch als ein natürlichsprachliches Ver-

633) Siehe hierzu Holsapple, Joshi (1999), S. 7 ff., und Holsapple, Singh (2003), S. 215 ff.

634) Vgl. Bouaud, Bachimont et al. (1994), Bouaud, Bachimont et al. (1995).

ständnissystems bezeichnet wird, auf der Basis konzeptueller Graphen.[635] Nachfolgend werden die vier Prinzipien kurz erläutert:[636]

1. Gleichheitsprinzip
   Eine Unterklasse muss vom selben Typ wie die Oberklasse sein.[637]

2. Genauigkeitsprinzip
   Eine Unterklasse muss von der Oberklasse unterscheidbar sein. Die Unterschiede zusammen mit der Definition der Oberklasse bilden die notwendigen und hinreichenden Bedingungen für die Definition der Unterklasse.

3. Gegensatzprinzip
   Die Unterklassen einer Oberklasse sollen disjunkt sein, d. h. alle Unterklassen müssen paarweise inkompatibel sein.

4. Prinzip der Einzigartigkeit semantischer Achsen
   Die Unterklassen einer Oberklasse sollen anhand einer gemeinsamen Dimension (Typ) oder Achse definiert werden, um sich von der Oberklasse zu unterscheiden. Eine gemeinsame Achse soll jeweils nur einmal in der Ontologie Berücksichtigung finden.

Jones, Bench-Capon et al. merken an, dass das hier kurz umrissene Vorgehensmodell einen idealisierten Blick auf Taxonomien als Möglichkeit zur Repräsentation von Wissen über Domänen wirft. Sie zweifeln grundsätzlich an der Anwendbarkeit taxonomischer Repräsentationen bei vielen Domänen.[638]

5.1.2.3.2.2 KACTUS-Ansatz

Die Autoren Schreiber, Wielinga und Jansweijer des Projekts KACTUS (modelling Knowledge About Complex Technical systems for multiple USe), das im Rahmen des Forschungsförderungsprogramms ESPRIT der Europäischen Union durchgeführt wurde, gehen von einer bereits bestehenden Wissensbasis, die aus den unterschiedlichsten Quellen (etwa aus Datenbankschematas) zusammengesetzt sein kann, aus. Durch Abstraktion wird aus dieser Wissensbasis eine generelle Ontologie entwickelt.[639]

---

635) Vgl. Bouaud, Bachimont et al. (1994), S. 2. Zum Komplex der konzeptuellen Graphen siehe auch Kapitel 4.1.3.2.4, S. 83, und zur Einführung siehe Sowa (2000), S. 476 ff.

636) Vgl. Bouaud, bachimont et al. (1995), S. 5.

637) In dem hier verwendeten Sinn wird mit Typ ein gemeinsames Merkmal, das eine Klasse konstituiert, bezeichnet. Falls bspw. die Oberklasse „familie" zum Typ „mensch" gehört, dann muss die mögliche Unterklasse „kind" ebenfalls zu diesem Typ gehören.

638) Vgl. Jones, Bench-Capon et al. (1998), S. 9. Diese Zweifel werden vom Verfasser geteilt.

639) Vgl. Sure (2003), S. 220.

KACTUS[640] ist ein applikationsgetriebenes[641] Projekt zur Wiederverwendung von Wissen technischer Domänen und beschäftigt sich hierzu insbesondere mit der Ontologieentwicklung.[642] Die Autoren nehmen dabei den Standpunkt ein, dass jede vorhandene Applikation bereits die Entwicklung einer „kleinen" Ontologie darstellt.[643] Aus diesen Applikationen sind die Ontologien anschließend zu extrahieren. Dabei werden mittels eines inkrementellen Vorgehens, bei dem ausgehend von „generellen" Ontologien zu „technischen", d. h. hinsichtlich des repräsentierten Wissens spezifischeren, Ontologien gelangt wird, folgende Schritte unternommen:[644]

1. Spezifizierung der Applikation
   Die Spezifizierung der Applikation legt zum einen den Kontext, in dem die Applikation angewendet wird, und zum anderen den „Blick", der durch die Systemkomponenten der Applikation in einer Domäne eingenommen wird, fest. Es wird eine Liste von Konzepten, die in der Ontologie Berücksichtigung finden sollen, aufgestellt.

2. Vorläufiges Design
   Auf Basis der in der Phase der Spezifizierung ermittelten Liste von Konzepten wird eine präliminare Ontologie entwickelt. Hierzu ist es notwendig, den Top-Level der Ontologie festzulegen.

3. Ontologieverfeinerung und -strukturierung
   In einem weiteren Schritt wird die präliminare Ontologie verfeinert und in eine endgültige Form gebracht, die von den Ontologieentwicklern als vollständig angesehen wird.

Eine „vollständige, große" Ontologie wird aus mehreren „kleineren" Ontologien entwickelt. Hierzu wird insbesondere ein „Mapping", d. h. die explizite Verbindung im Sinne einer Ver-

---

640) KACTUS gilt als Nachfolgeprojekt zu CommonKADS, das ebenfalls durch Mittel der Europäischen Union ermöglicht wurde. Siehe hierzu auch Kapitel 5.4.3.1, S. 187 f.

641) „Applikationsgetrieben" bedeutet in diesem Fall grob vereinfacht, dass vorhandene Applikationen als Ausgangspunkt einer Ontologieentwicklung dienen. Dies entspricht einer Umkehrung der üblichen Vorgehensweise, in der eine ontologiebasierte Applikation entwickelt werden soll. Eine Applikation ist ein Computerprogramm, das einem bestimmten Zweck dient. Es wird synonym für den Begriff „Anwendungsprogramm" verwendet, weil sich aus dem bedeutungsgleichen englischen Begriff „Application" im allgemeinen deutschen Sprachgebrauch auch die Bezeichnung „Applikation" als direkte Übersetzung durchgesetzt hat. Der Begriff „Anwendungsprogramm" steht im Gegensatz zum Betriebssystem und allen System- und Hilfsprogrammen, wie etwa Werkzeugen zur Softwareerstellung, die lediglich den Betrieb einer Software ermöglichen, aber noch keinen darüber hinausgehenden Nutzen für den Benutzer bringen, der nicht selbst Entwickler ist.

642) Vgl. Schreiber, Wielinga et al. (1995), S. 1; van Heijst, Schreiber et al. (1997), S. 183 ff.

643) Verwendet man das Verständnis zur Existenz von leichtgewichtigen Ontologien, dann scheint es möglich, dieser Auffassung uneingeschränkt zu folgen, weil sich je nach Betrachtungswinkel in jeder Anwendung Klassen bspw. von Funktionen finden lassen, die in einer Taxonomie dargestellt werden können. Auf weitere Ausführungen hierzu wird an dieser Stelle verzichtet, weil diese Ausführungen zu weit von der Problemstellung dieser Arbeit wegführen würden.

644) Vgl. Studer, Benjamins et al. (1998), S. 188, zum inkrementellen Vorgehen und Fernández López (1999), S. 4-7; Gómez-Pérez, Fernández-López et al. (2004), S. 124 f., zu den drei Schritten.

schmelzung, der eingehenden Ontologien notwendig. Es lassen sich zwei Arten von Mapping-Funktionen zwischen Ontologien unterscheiden:[645]

1. Zum einen kommt es zu keiner Änderung hinsichtlich der Semantik der verwendeten Konstrukte in der „gemappten" Ontologie.

2. Zum anderen kommt es zu einer Änderung der Semantik, wenn eine Neu-Interpretation der Konstrukte (bspw. aufgrund von Doppelnennungen[646]) notwendig wird.

Im vorangegangenen Kapitel 5.1.2.3.1 wurden Vorgehensmodelle untersucht, die insbesondere eine Ontologieentwicklung aus dem „Nichts" ermöglichen sollen, d. h. es wird von keiner vorherigen Konzeptualisierung ausgegangen. KACTUS verhält sich hierzu entgegengesetzt und wurde deshalb nicht in dieses Kapitel 5.1.2.3.1 aufgenommen.[647] Es wird bei KACTUS davon ausgegangen, dass Applikationen, die einer Konzeptualisierung zur Realisierung einer Ontologie bedürfen, dem Entwickler bereits vorliegen und er lediglich diese Konzeptualisierung quasi „zurückführen" muss.

5.1.2.3.2.3 SENSUS-Ansatz

Im SENSUS-Ansatz, vorgestellt von Swartout, Patil, Knight und Russ[648], unterscheiden die Autoren zwischen *Domänen-Ontologien* und *Theorie-Ontologien*. Während eine Theorie-Ontologie eine Menge von Konzepten bereithält, die einen allgemeingültigen Aspekt der Welt abbilden (z. B. Zeit oder Raum), werden bei einer Domänen-Ontologie Konzepte abgebildet, die eine bestimmte Domäne beschreiben.[649] Theorie-Ontologien sind erwartungsgemäß weniger umfangreich als Domänen-Ontologien, die leicht tausende Konzepte berücksichtigen können. Der SENSUS-Ansatz berücksichtigt lediglich Domänen-Ontologien.

SENSUS stellt eine natürlichsprachliche („natural language based") Ontologie dar, die in ihrer Endversion über 50.000 Konzepte beinhaltet.[650] Sie ist das Ergebnis der Verbindung von

645) Vgl. Schreiber, Wielinga et al. (1995), S. 6.

646) Wenn bspw. eine Ontologie zur Domäne der Geldinstitute und eine Ontologie zur Domäne der Sitzgelegenheiten gemappt werden, so wird es höchstwahrscheinlich zu einer Doppelnennung des Konzepts „Bank" kommen. In der vollständigen großen Ontologie erweitert sich die Semantik des Konzepts somit auf beide Bedeutungen aus den jeweiligen ursprünglichen Domänen oder es wird in einer Ontologie möglicherweise bei einer Neuinterpretation notwendig, eindeutige Konzepte, die jeweils nur einmal Verwendung finden, festzulegen und somit die alte Verwendungsweise abzuschaffen.

647) Diese Entscheidung wird bspw. auch durch Gómez-Pérez, Benjamins (1999) gestützt. Die Autoren zählen dort KACTUS nicht zu den Vorgehensmodellen für die Ontologieentwicklung (S. 1.4 f.), sondern zu Applikationen, die Ontologien verwenden (S. 1.6 f.).

648) Vgl. Swartout, Patil et al. (1996); Swartout, Patil et al. (1997).

649) Die von den Autoren vorgenommene Unterscheidung wird im Sinne einer Arbeitshypothese von diesen verwendet, denn es erfolgt keine ausreichende Definition hinsichtlich der Eindeutigkeit der Zuordbarkeit. Die Autoren erwähnen lediglich, dass Theorie-Ontologien zu einer höheren Abstraktion tendieren als Domänen-Ontologien (vgl. Swartout, Patil et al. (1997), S. 139). Aus der abstrakteren Sichtweise ergibt sich möglicherweise auch der Grund für die eigenwillige Bezeichnung "Theorie-Ontologien", die an dieser Stelle eher irreführend wirkt und möglicherweise mit "Common-Sense-Ontologien" oder „Meta-Ontologien" gelungener erfolgt wäre.

650) Vgl. Swartout, Patil et al. (1997), S. 138.

Penman Upper Model, Ontos, WordNet und Begriffssammlungen aus elektronischen Wörterbüchern.[651]

Bei der Entwicklung von „neuen“ Domänen-Ontologien soll SENSUS als Ausgangsbasis genutzt werden. Hierzu werden zur Erstellung einer spezifischen Domänen-Ontologie „Saat“-Konzepte (*„seed“ terms*) als repräsentativ selektiert und von Hand mit SENSUS verlinkt. Alle Konzepte ausgehend von den Saat-Konzepten hin zu den Wurzel-Konzepten (*„root“ terms*) werden in der zu erstellenden Domänen-Ontologie berücksichtigt. Hinzu kommen relevante semantische Kategorien und ganze Unterkonzeptbäume, die eine besondere Bedeutung für die Verbindung von Saat- und Wurzel-Konzepten darstellen. Die verbleibenden SENSUS-Konzepte werden von den Entwicklern der Ontologie als irrelevant verworfen, um Speicherplatz zu sparen und die Effizienz durch schnellere Antwortzeiten des Anwendungssystems zu erhöhen.

5.1.2.3.2.4 ONIONS-Ansatz

Der ONIONS-Ansatz (ONtologic Integration Of Naïve Sources) resultiert aus dem Interesse, bereits existierende Ontologien zu integrieren. Er wurde ab 1992 entwickelt[652] und gehört zu den elaboriertesten Ansätzen zur Integration von Ontologien.[653] Die Ziele waren:

- eine „gut“ abgestimmte Menge von generischen Ontologien zu entwickeln,
- die Integration relevanter Domänen-Ontologien in eine formale, konzeptionell befriedigende Ontologie voranzutreiben und
- ein explizites Nachverfolgen der Ontologieentwicklung zu ermöglichen.[654]

Der Ansatz beinhaltet ein allgemeines Vorgehen zur Interpretation von fachsprachlichen Definitionen innerhalb spezifischer Domänen-Ontologien, die zusammengeführt werden sollen. Darauf aufbauend beinhaltet der Ansatz ein allgemeines Vorgehen zur anschließenden Erstellung einer neueren und offeneren Domänen-Ontologie, die die zu integrierenden Domänen-Ontologien vereint. Das Vorgehen gliedert sich in sechs Schritte:

1. Extrahieren von Quell-Konzepten aus den vorhandenen Domänen-Ontologien, die als besonders relevant für eine Domäne gelten,
2. Feststellen der lokalen Definitionen der Quell-Konzepte, d. h. der festgelegten Verwendungsweise bspw. innerhalb einer Taxonomie, in den zu integrierenden Domänen-Ontologien,

651) Vgl. Knight, Luk (1994), S. 773 ff. Vgl. zum Penman Upper Model Bateman, Kasper et al. (1989) und Bateman, Magnini et al. (1994). Vgl. zu ONTOS Carlson, Nirenburg (1990). Vgl. zu WordNet Fn. 516.

652) Vgl. Gangemi, Pisanelli et al. (1998), S. 163. Vgl. Ferner Gangemi, Steve et al. (1996).

653) Vgl. Gómez-Pérez, Fernández-López et al. (2004), S. 164.

654) Vgl. Gangemi, Pisanelli et al. (1998), S. 163.

3. Anreicherung der lokalen Definitionen in Bezug auf eine generelle Theorie, die es ermöglicht, die einzelnen lokalen Definitionen zu integrieren,

4. Weiterentwicklung dieser generellen Theorie zur Bildung von Top-Level-Konzepten,

5. Zusammenführung von lokalen Konzepten und Top-Level-Konzepten zu einer gemeinsamen Ontologie und

6. Formalisierung der integrierten Ontologie.[655]

ONIONS fokussiert hierbei vor allem auf Probleme bei der Integration von Ontologien, wie sie häufig mit Anwendung eines Abbruch- und eines Relevanz-Kriteriums einher-gehen. Das Abbruch-Kriterium beschäftigt sich mit dem notwendigen Detaillierungsgrad gemäß den Anforderungen einer Konzeptualisierung, die bspw. durch Benutzer und Entwickler vorgegeben werden. Das Relevanz-Kriterium beschäftigt sich mit der Frage, wie festgestellt werden kann, was als konzeptionell relevant anerkannt wird. Wie Jones, Bench-Capon et al. feststellen, bleiben die Autoren jedoch hier ausreichende Antworten schuldig.[656]

5.1.2.3.2.5 OntoClean-Ansatz

Der OntoClean-Ansatz wurde von Guarino und Welty vorgestellt.[657] Er dient insbesondere der Validierung von Taxonomien, indem der Ansatz unangemessene und inkonsistente Modellierungen in einer Taxonomie aufdeckt.[658] Der OntoClean-Ansatz verwendet formale Konzepte aus der analytischen Philosophie, die so allgemeinen Charakter besitzen, dass sie in jeder Ontologie Verwendung finden können. Zu diesen Konzepten gehören „Rigidity", „Identity" und „Unity".[659] Mit diesen Konzepten wird ein Satz von Meta-Eigenschaften definiert, der genutzt wird, um relevante Teile von Konzepten, Attributen und Relationen mit ihren intendierten Bedeutungen abzubilden. Aus diesem Vorgehen ergeben sich Erkenntnisse hinsichtlich der Bedeutung der verwendeten Konzepte und damit ihrer Anordnung in einer Taxonomie. Das Vorgehen besteht im Wesentlichen aus dem Korrigieren von unangemessenen oder

655 An dieser Stelle wird deutlich, dass die Autoren eine abweichende Definition von Ontologien verwenden. In der Arbeitsdefinition dieser Arbeit liegt erst nach der Formalisierung eine Ontologie vor.

656) Vgl. Jones, Bench-Capon et al (1998), S. 9.

657) Vgl. Guarino, Welty (2002) und Guarino, Welty (2004).

658) Vgl. Guarino, Welty (2002), S. 61 und S. 63.

659 Vgl. Guarino, Welty (2002), S. 61 ff. Ins Deutsche lassen sich die Eigenschaften in „Rigidität", „Identität" und „Vollständigkeit" übersetzen. Eine Eigenschaft eines Konzepts gilt als rigide, wenn sie als essentiell für alle Instanzen des Konzepts angesehen wird (bspw. müssen für die Eigenschaft „menschlich zu sein" alle Instanzen notwendig menschlich sein). „Identität" beschäftigt sich mit der Möglichkeit, Konzepte einer Welt als gleich (oder unterschiedlich) zu erkennen und „Vollständigkeit" beschäftigt sich mit der Möglichkeit, alle Konzepte einer Welt zu identifizieren, die ein bestimmtes (Ober-)Konzept vollständig konstituieren.

inkonsistenten Modellierungen innerhalb der vorhandenen Konzepte einer Taxonomie, die mit Hilfe des Satzes der Meta-Eigenschaften aufgedeckt werden.[660]

Der Ansatz findet erst bei einer bestehenden Ontologie Verwendung. Die Entwicklung einer „neuen“ Ontologie aus dem Nichts wird nicht unterstützt.

5.1.3 Abgrenzung von anderen Instrumenten

In der wirtschaftswissenschaftlichen Forschung lassen sich Vorgehensmodelle, Branchenreferenzmodelle und Softwarereferenzmodelle unterscheiden:[661]

Vorgehensmodelle beschreiben Projektabläufe, wie bspw. die Durchführung eines Projekts zum Business Process Reengineering, die Einführung einer Standardsoftware oder die Durchführung einer ISO-Zertifizierung.

Branchenreferenzmodelle dokumentieren sowohl Prozesse als auch Daten- und Funktionsstrukturen, die typisch für eine bestimmte Wirtschaftsbranche sind. Branchenmodelle beziehen sich z. B. auf operative Geschäftsprozesse wie Logistik, Produktentwicklung und Rechnungswesen. In ARIS (Architektur Integrierter Informationssysteme), das die methodische Grundlage für die Branchenreferenzmodelle von Scheer darstellt[662], stehen z. B. Modelle für die Branchen Papierindustrie, Chemische Industrie, Maschinenbau, Anlagenbau und Energie zur Verfügung.[663]

Softwaremodelle bilden die typischen Prozesse, Funktions- und Datenstrukturen von (Standard-)Softwaresystemen ab. Softwaremodelle können bei der Auswahl und Einführung einer Software zum Vergleich zwischen Anforderungen und angebotenen Funktionalitäten der Software sowie zur spezifischen Anpassung von Standardsoftware an (bspw.) die spezifischen betrieblichen Gegebenheiten „vor Ort“ eingesetzt werden.[664]

660 Den gesamten Ansatz an dieser Stelle wiederzugeben wird vom Verfasser als nicht zielführend für diese Arbeit erachtet. In Guarino, Welty (2004) findet sich ein anschauliches Beispiel ab S. 157 zur Anwendung von OntoClean.

661) Vgl. Scheer (1997), S. 7; Lang (1997) S. 21 ff.; Reiter (1999), S. 46 ff.

662) Vgl. Reiter (1997), S. 34.

663) Vgl. http://www.ids-scheer.de/germany/31626, Zugriff am 18.03.2007.

664) Vgl. Scheer (1997), S. 9.

5.1.4 Anwendung von Vorgehensmodellen

5.1.4.1 Durchführung der Entwicklung von Vorgehensmodellen

Mit Hilfe von Formal-Sprachen können Vorgehensmodelle formal beschrieben werden.[665] Dies ermöglicht, die Modelle automatisch auf Konsistenz zu prüfen.[666] Alternativ zur formalen Darstellung kann eine semi-formale Darstellungsform gewählt werden, die eine Anwendbarkeit des Vorgehensmodells (im Sinne einer Nachvollziehbarkeit durch den Benutzer) wesentlich erleichtert, weil in diesem Fall natürlichsprachliche Elemente Teil des Vorgehensmodells sind und somit zur Erleichterung des „Verstehbarkeit" beitragen können.

Es existieren unterschiedliche Modellierungssprachen, um Vorgehensmodelle formal zu beschreiben.[667]

Für die Abbildung von Abläufen und damit für die Repräsentation von prozeduralem Wissen kann man grundsätzlich zwischen aktivitätsorientierten, ereignisorientierten und objektorientierten Methoden unterscheiden. Entsprechend ihrer Namensgebung betonen sie unterschiedliche Merkmale bei der Prozessdarstellung.

Aktivitätsorientierte Methoden bilden die Aktivitäten eines Prozesses, deren Verzweigungen und zeitliche Anordnung ab, können aber keine Ereignisse, d. h. das „Eingetretensein" eines geforderten Zustands, zwischen den einzelnen Aktivitäten darstellen. Beispielhaft für aktivitätsorientierte Methoden sind Programmablaufpläne.

Ereignisorientierte Methoden stellen mit Hilfe von Ereignissen, Zuständen und Aktivitäten den konditionalen (d. h. den zeitlich-sachlogischen) Prozessablauf dar. Bei den ereignisorien-

665) Zur Formalisierung von Vorgehensmodellen siehe Verlage (1998a), S. 61 ff.

666) Verlage nennt bspw. Appl/A (Ada Process Programming Language with Aspen), MSL (Marvel Strategy Language), MVP-L (Multi View Process Modeling Language), SLANG (Spade Language), Statemate und Tempo (vgl. Verlage (1998b), S. 82 ff.).

667) Vgl. Verlage (1998b), S. 76 ff., und bedingt auch Schütte (1998), S. 92 ff. Letzterer formuliert eine Sprachauswahl für die Beschreibung von Informationssystemen, die ausgewählten Sprachen können jedoch auch für die Zwecke der Beschreibung von Vorgehensmodellen genutzt werden.

tierten Methoden sind vor allem Ereignisgesteuerte Prozessketten (EPK) und Petri-Netze von herausragender Bedeutung.[668]

Objektorientierte Methoden stellen Objekte in den Mittelpunkt der Modellierung. Die Darstellung des Prozessablaufs erfolgt mit Hilfe von Nachrichten (auch als Transaktionen bezeichnet), die zwischen den Objekten ausgetauscht werden, wie bspw. im Semantischen Objektmodell (SOM)[669] angewendet.

In dieser Arbeit wird eine Sprache verwendet, die grundsätzlich in der Lage ist, alle drei Kategorien abzudecken. Es handelt sich dabei um die Unified Modeling Language (UML)[670], sie wird zur Modellierung des OntoFMEA-Vorgehensmodells eingesetzt. Auf eine weit reichende Analyse möglicher weiterer Modellierungssprachen für Vorgehensmodelle wurde aus pragmatischen und den vorliegenden Anforderungen[671] entsprechenden Überlegungen ver-

668) Für (erweiterte) Ereignisgesteuerte Prozessketten (EPKn) ist die Gliederung eines Prozesses in Ereignisse und Funktionen typisch. Eine Funktion entspricht dabei einer (betriebswirtschaftlichen) Aktivität an einem oder mehreren Objekten (bspw. Datenbank, Kundendaten, Organisationseinheit). Ereignisse lösen Funktionen aus und durchgeführte Funktionen erzeugen wiederum Ereignisse. Jeder Prozess beinhaltet mindestens ein Startereignis und ein Endereignis. Funktionen werden mit einem Substantiv (oftmals für das bearbeitete Objekt) und einem die Aktivität charakterisierenden Verb benannt. Ereignisse werden demgegenüber mit einem Substantiv (oftmals für das bearbeitete Objekt) und einem Verb zur Formulierung des Zustands benannt. Es können nur Funktionen und Ereignisse abwechselnd miteinander verbunden werden. Neben direkten Verbindungen kommen auch Konnektoren zum Einsatz. Hauptsächlich lassen sich drei Arten von Konnektoren unterscheiden. Diese können auch kombiniert angewendet werden. Es werden Konnektoren verwendet, die „konjunkt“, „adjunkt“ und „disjunkt“ wirken, das heißt sie wirken als „und“, „inklusives oder“ und als „exklusives oder“. Verschiedene Prozesse (eEPKs) können mit Prozesswegweisern, die in etwa wie Platzhalter wirken, verbunden werden. Zwangsläufig entsprechen dann die Endereignisse des vorlaufenden Prozesses den Anfangsereignissen des nachfolgenden (verwiesenen) Prozesses, dies entspricht einer vertikalen Dekomposition eines Gesamtprozesses (Hierarchisierung zueinander in Verbindung stehender Teilprozesse). Des Weiteren gibt es die horizontale Segmentierung, die miteinander über Prozessschnittstellen verbundene, einzelne Prozessmodelle nebeneinander darstellt.
Petri-Netze (PN) sind benannt nach C. A. Petri, der sie 1962 in seiner Dissertation als Modelle für nebenläufige (es besteht die Möglichkeit des parallelen Verlaufs, beliebiger Reihenfolgen und „Verschachtelungen“) Prozesse eingeführt hat. Ein abgebildetes System wird insbesondere durch seine passiven Elemente (Stellen) und seine aktiven Elemente (Transitionen), die in einem kausal-logischem Zusammenhang stehen, gekennzeichnet. In der Wissenschaft sind verschiedene Varianten von Petri-Netzen bekannt, wie Stelle/Transition-Netze, farbige Petri-Netze, hierarchische Petri-Netze, Bedingungs/Ereignisnetze, stochastische und zeitbehaftete Petri-Netze, (erweiterte) Prädikate/Transitions-Netze und Kanal/Instanzen-Netze. Petri-Netze eignen sich insbesondere zur Repräsentation von Wissen, bei dem diskrete Ereignisse auftreten, die von diskreten Objekten verbraucht oder erzeugt werden. Vgl. zu eEPK Seidlmeier (2002), S. 70 ff., und zu Petri-Netzen Baumgarten (1990); Zelewski (1995), S. 25 ff.

669) Vgl. Ferstl, Sinz (1993), S. 589 ff., und Ferstl, Sinz (1994).

670) Vgl. OMG (2003) und Rumbaugh, Jacobson et al. (2005). Für eine Verwendung der UML als Repräsentationssprache für Ontologien siehe Guizzardi, Wagner et al. (2004). Stuckenschmidt und van Harmelen gehen in ihren Äußerungen sogar so weit zu behaupten, dass die UML selbst eine Ontologie darstellt (vgl. Stuckenschmidt, van Harmelen (2005), S. 29). Dieser Auffassung schließt sich der Verfasser jedoch nicht an, weil bspw. das UML-Metamodell eine Verwendung des Konzepts „property“ gemäß DAML nicht erlaubt. Erst nach einer Anpassung, wie etwa in Baclawski, Kokar et al. (2002), S. 151 ff., vorgeschlagen, wird es möglich, Ontologien mittels UML darzustellen.

671) Anforderungen an die Modellierungssprache stellen sich für die wirtschaftswissenschaftliche Problemstellung in Form der Gütekriterien des Kapitels 5.3.3.2, S. 170 ff.

zichtet. Insbesondere die folgenden Gründe fanden bei der Entscheidung für UML Berücksichtigung:[672]

1. Um eine Vereinheitlichung der verwendeten Notationen über die gesamte Arbeit zu erreichen, wird nach Möglichkeit auf Standards zurückgegriffen. Bereits an anderer Stelle dieser Arbeit wird die UML verwendet.[673]

2. Im Prinzip ist das zu modellierende OntoFMEA-Vorgehen aufgrund seiner geringen Komplexität in den meisten Sprachen, die Abläufe zu modellieren vermögen, darstellbar.[674]

3. Bei der UML handelt es sich um einen offenen Standard, der frei zugänglich ist.

4. Es handelt sich um eine Sprache, die auch eine grafische (semi-formale) Notation ermöglicht und so die „Einfachheit" der Nachvollziehbarkeit berücksichtigt.

5. Die UML hat eine weite Verbreitung im Bereich der Softwareentwicklung gefunden[675], und auch für die (Geschäfts-)Prozessmodellierung wird ihr großes Potential zugeschrieben.[676]

6. Dem Verfasser stand ein Software-Werkzeug zur Verfügung, das die Modellierung mittels UML unterstützt.[677]

Die UML ist mit der Entwicklung des objektorientierten Paradigmas in der Softwareentwicklung entstanden.[678] Arbeiten von Booch, Jacobson und Rumbaugh[679] begründeten die UML,

672) Puri verwendet ebenfalls die UML, um seine semantische Funktionsmodellierung darzustellen (vgl. Puri (2003), S. 82). Er hebt dabei hervor, dass die UML ermöglicht, Entwurf und Entwicklung sowohl von Softwaremodellen als auch von Nicht-Softwaremodellen auf einer gemeinsamen Basis zu diskutieren (ebenfalls S. 82). Leutsch verwendet die UML, um seine prozedurale Wissensunterstützung eines Konstruktionsprozesses in einem Datenmodell darzulegen (vgl. Leutsch (2002), S. 119 ff.). Noack stellt einen Ansatz vor, der in einem durchgängigen Entwicklungsprozess ein Vorgehensmodell vorsieht, das anfänglich als EPK vorliegende Geschäftprozessmodelle in die UML-Notation überführt, weil diese als objekt-orientierte und damit aktuelle Softwareentwicklung für gebotener gehalten wird (vgl. Noack (2000), S. 6 und S. 12 f.).

673) Siehe S. 243.

674) Dieser Standpunkt führt zum selben Ergebnis wie die Meinung von Verlage, der die Suche nach einer für sämtliche denkbaren Anwendungsfälle „optimalen" Modellierungssprache als nicht sinnvoll erklärt, weil etwaige Anforderungen zu unterschiedlich ausfallen (vgl. Verlage (1998b), S. 79).

675) Vgl. Rumbaugh, Jacobson et al. (2005), S. 6; Borrmann, Komnik et al. (2002), S. 31; Leutsch (2002), S. 175.

676) Vgl. Eriksson, Penker (2000), S. XV f.

677) Es handelt sich hierbei um die Software MS Visio 2003 der Microsoft Deutschland GmbH (vgl. http://office.microsoft.com/de-de/FX010857981031.aspx, Zugriff am 15.08.2005).

678) Vgl. Burkhardt (1997), S. 1 ff.

679) Vgl. Burkhardt (1997), S. 1 ff.

deren Standard heute durch die OMG[680] vorgegeben wird. UML wird in vielen Branchen[681] und für verschiedenste Aufgaben als Dokumentationssprache[682] verwendet.

5.1.4.2 Einsatzgebiete von Vorgehensmodellen

Seit geraumer Zeit finden Vorgehensmodelle praktischen Einsatz.[683] In allen Bereichen der Softwareentwicklung, in denen eine systematische Vorgehensweise vorteilhaft ist, finden Vorgehensmodelle Verwendung. Deshalb wird an dieser Stelle nur ein kleiner Ausschnitt wiedergegeben, um die breite Anwendung lediglich anzudeuten. Es werden jeweils drei Einsatzgebiete exemplarisch genannt.

Für den Bereich des Software Engineering finden sich Einsatzgebiete für Vorgehensmodelle wie

- Banken [Fischer, Grünbacher et al. (1999); Noack (2000)],
- Telekommunikation [Dimitrov, Schmietendorf et al. (2000)] und
- Multimedia [Kopka (2000)].

Für den Bereich des Knowledge Engineering finden sich Einsatzgebiete für Vorgehensmodelle wie

- Produktion [Ranjan, Glassey et al. (2002), Moynihan, Bowers et al. (2002)],
- Ingenieurwesen [Ngai, Chow (1999)] und
- E-Learning [Bertin, Buciol et al. (1998)].

Für den Bereich des Ontology Engineering finden sich Einsatzgebiete für Vorgehensmodelle wie

- Kompetenzmanagement [Apke, Bremer et al. (2004); Lau, Sure (2002)],
- Ingenieurwesen [Fernández López, Gómez-Pérez et al. (1999)] und
- Management-Informationssysteme [Fox, Grüninger (1998); Tsou, Kao (2003); Uschold, King et al. (1998)].

---

680) Vgl. OMG (2003).

681) Vgl. Booch, Rumbaugh et al. (1999), S. 17.

682) Vgl. Booch, Rumbaugh et al. (1999), S. 16.

683) Zu einer Genealogie von Vorgehensmodellen siehe Bremer (1998), insbesondere S. 31.

5.1.4.3 IT-Unterstützung

Eine IT-Unterstützung für Vorgehensmodelle lässt sich unterscheiden in eine IT-Unterstützung bei der *Entwicklung* eines Vorgehensmodells und eine IT-Unterstützung bei der *Anwendung* eines Vorgehensmodells.

Am Markt finden sich zahlreiche Software-Anwendungen zur Entwicklung von Vorgehensmodellen. Dabei lassen sich i. d. R. sämtliche Modellierungswerkzeuge der Softwareentwicklung zur Vorgehensmodellentwicklung heranziehen. Weil es letztendlich für das mit dieser Arbeit angestrebte OntoFMEA-Vorgehensmodell unerheblich erscheint, mit welcher Software es erstellt wurde, wird auf eine Marktübersicht an dieser Stelle verzichtet.[684]

Als noch schwieriger zu erfassen erweist sich der Bereich der IT-Unterstützung bei der Anwendung eines Vorgehensmodells. In der Regel handelt es sich bei der IT-Unterstützung um an die jeweiligen Bedürfnisse der Benutzer angepasste Systeme. Eine Anpassung erfolgt dabei bspw. hinsichtlich Betriebsgröße oder Branche, für die Software entwickelt wird. Die Abbildung 33 verdeutlicht dies exemplarisch mit einem Bildschirmabzug der Web-Anwendung des KOWIEN-Vorgehensmodells, das für die Entwicklung von ontologiebasierten Kompetenzmanagementsystemen für kleine und mittelgroße Unternehmen vornehmlich des Maschinenbaus entwickelt wurde.[685]

Oftmals findet sich als ein kleinster gemeinsamer Nenner eine IT-Unterstützung in Form einer elektronischen „Bibliothek“, die über das Internet verfügbar gemacht wird, mit Hinweisen, Vorlagen und weiteren unterstützenden Werkzeugen.[686]

684) Eine Marktübersicht zu Modellierungswerkzeugen würde bei weitem den Rahmen dieser Arbeit sprengen, ohne einen substantiellen Beitrag zur Problemstellung der Arbeit leisten zu können. Dem Verfasser stand als proprietäres Werkzeug MS Visio der Microsoft Deutschland GmbH zur Verfügung. Dieses wurde dann auch zur Modellierung eingesetzt.

685) Vgl. zur Web-Anwenung des KOWIEN-Vorgehensmodells Apke, Dittmann (2004), S. 99 ff. Für weitere Informationen zum Forschungsverbundprojekt KOWIEN (Kooperatives Wissensmanagement in Engineering-Netzwerken) siehe http://www.kowien.uni-essen.de, Zugriff am 4.3.2007, und Zelewski, Alan et al. (2005).

686) So zum Beispiel bei Ruth (2000), S. 146 und Dimitrov, Schmietendorf et al. (2000), S. 61 ff.

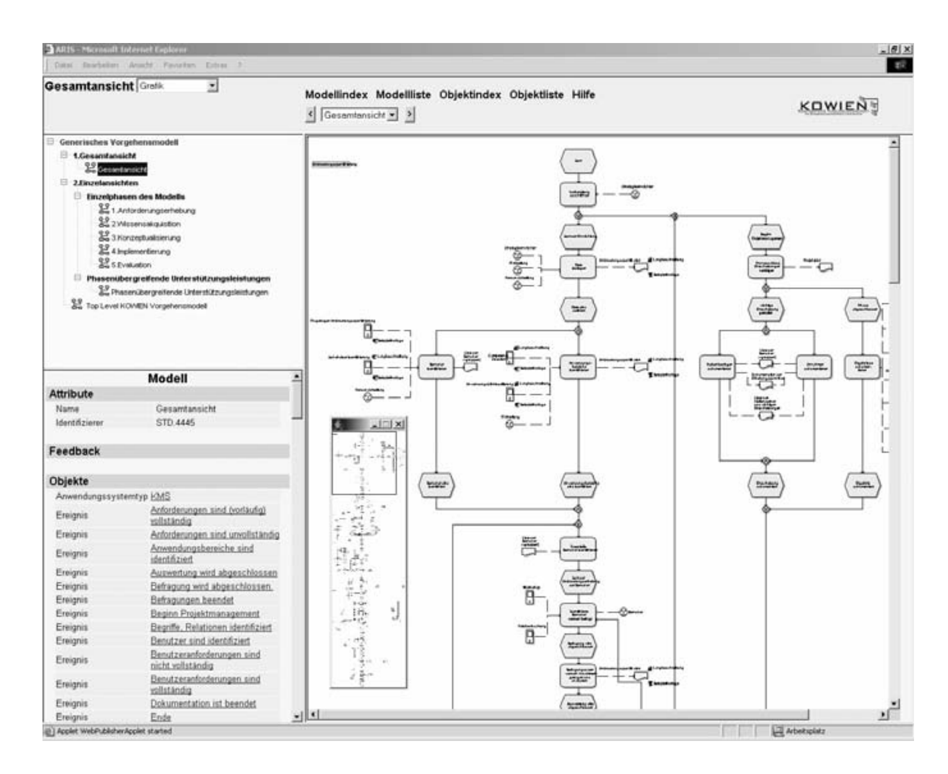


Abbildung 33: Web-Anwendung KOWIEN-Vorgehensmodell[687]

687) Vgl. Apke, Dittmann (2004), S. 102.

5.2 Repräsentation von Wissen in Vorgehensmodellen

Die Darstellung eines Vorgehensmodells beinhaltet das Wissen eines Vorgehensmodells, d. h., das Wissen über ein bestimmtes Vorgehen hängt unmittelbar von dessen Darstellung ab. In den Modellierungsprimitiven der verwendeten Repräsentationssprache determiniert sich die Repräsentationsmöglichkeit eines (Vorgehens-)Modells, d. h., wenn bspw. Artefakte in einem Vorgehensmodell Berücksichtigung finden sollen, so wird es notwendig, eine Repräsentationssprache zu verwenden, die die explizite Darstellung von Artefakten vorsieht.

Zur Darstellung des OntoFMEA-Vorgehensmodells wird auf die Spezifikation von Aktivitätsdiagrammen[688] innerhalb der UML zurückgegriffen, weil diese Diagramme als „…*Technik prozedurale Logik, Geschäftsprozesse und Arbeitsabläufe beschreiben*“[689] und somit für die Darstellung des OntoFMEA-Vorgehens geeignet sind:

Start- und Endzustand: Jedes Vorgehensmodell beginnt mit genau einem *Startzustand* und jedes Vorgehensmodell endet mit genau einem *Endzustand*. Ein Startzustand wird über einen schwarzen Kreis dargestellt. Ein Endzustand wird über einen schwarzen Kreis mit einer doppelten äußeren Linie dargestellt (siehe auch Abbildung 34).

Aktion: *Aktionen* werden durch ein grafisches Element mit geraden Seiten und konvex geformten Ecken dargestellt. Eine Aktion entspricht einem Vorgehensschritt im Vorgehensmodell und besitzt einen Aktionsnamen. Die Erläuterung einer Aktion erfolgt in schriftlicher Form mittels Fließtext. Eine Aktivität (dargestellt in einem Aktivitätsdiagramm) umfasst mehrere Aktionen.

---

688) Aus dem Englischen „Activity Diagrams“ übersetzt. Zu Aktivitätsdiagrammen siehe Eriksson, Penker (2000), S. 40 ff.; Fowler (2004), S. 117 ff.; Kahlbrandt (2001), S. 178 ff., und die offizielle Spezifikation OMG (2003), S. 3-155 ff.
Für eine genaue Erläuterung sämtlicher Diagrammtypen sei auf die weiterführende Literatur Fowler (2004), Booch, Rumbaugh et al. (1999), Burkhardt (1997) und Oesterreich (2002) neben der offiziellen Spezifikation (OMG (2003)) verwiesen. Auch auf die Veränderung der Diagrammtypen und ihrer Einteilung bei der Umstellung der derzeit gültigen Version 1.5 der UML auf die Version 2.0 sei an dieser Stelle hingewiesen (siehe hierzu für UML 1.5 OMG (2003), S. 2-6 ff., und für UML 2.0 Fowler (2004), S. 12).

689) Fowler (2004), S. 117. In dieser Arbeit werden im Wesentlichen die grafischen Elemente der Version 1.5 (Aktion, Transition, Objekt, Kontrollfluss, Objektfluss, Verzweigung, Aufspaltung, Vereinigung, Synchronisation, Start- und Endzustand) verwendet; zum einen, weil sie intuitiv verständlicher erscheinen als die grafischen Elemente der Version 2.0, und zum anderen, weil diese – zum Zeitpunkt der Erstellung dieser Arbeit – von der eingesetzten Modellierungssoftware lediglich unterstützt wurden. Drittens wurde die Version 2.0 noch nicht offiziell verabschiedet.
Jedoch lassen sich bereits mit der demnächst gültigen Version 2.0 einige Kritikpunkte an der „alten“ UML vermeiden. Bspw. brachte es die Weiterentwicklung der Aktivitätsdiagramme mit sich, dass diese nicht länger als eine Sonderform von Zustandsdiagrammen angesehen, sondern als eigenständiges Verhaltensdiagramm in das Meta-Modell der UML eingeordnet werden. Insgesamt wurde versucht, die Aktivitätsdiagramme zu einer „Petri-Netz-ähnlichen“ Modellierung zu entwickeln (vgl. Rumbaugh, Jacobson et al. (2005), S. 150 und S. 520, sowie Fowler (2004), S. 130 und S. 159). Deshalb wird eine Aktion nicht länger als ein (Aktions-)Zustand (activity state) mit einer internen Aktion angesehen (vgl. OMG (2003), S. 3-158), sondern vielmehr als ein (Aktivitäts-) Knoten (activity node), der eine Zustandsveränderung bewirkt oder einen Wert zurückliefert (vgl. Rumbaugh, Jacobson et al. (2005), S. 136).
Die hier im Folgenden vorgestellten Modellierungsprimitive verwenden den Zeichenvorrat und die Syntax der Version 1.5 und teilweise die Semantik der Version 2.0 (das Geschriebene gilt insbesondere für Objekte und Kanten).

Objekt: Ein *Objekt* ist ein Artefakt und wird durch ein grafisches Element mit geraden Seiten und spitzen Ecken dargestellt. Artefakte werden zur Durchführung des Vorgehensmodells gebraucht oder liegen anschließend als Ergebnisse vor. Es handelt sich bei den hier verwendeten Artefakten um Informationsträger (bspw. FMEA-Formular, Anforderungsspezifikation und Wissensträgerkarte). Ein Objekt besitzt einen Objektnamen. Die Erläuterung eines Objekts erfolgt ebenfalls in schriftlicher Form mittels Fließtext.

Kanten: *Kanten*[690] verbinden die Aktionen und Objekte eines Modells. Sie werden durch gerichtete Pfeile dargestellt. Eingehende Kanten lösen eine Aktion aus oder verändern ein Objekt („die Kante feuert"). Jede eingehende Kante kann allein eine nachfolgende Aktion auslösen[691]. Es werden zwei Arten von Kanten unterschieden. Zum einen gibt es einen Kontrollfluss (durchgezogene Linie) und zum anderen gibt es einen Objektfluss (gestrichelte Linie). Zwei Aktionen werden über einen Kontrollfluss miteinander verbunden. Der Kontrollfluss markiert den Übergang zum nächsten Vorgehensschritt. Ein Objekt wird immer mit einer Objektflusskante zu einer Aktion oder einem weiteren Objekt verbunden. Ein Objektfluss markiert den Einfluss eines Objekts auf die Durchführung einer Aktion oder den Einfluss einer durchgeführten Aktion auf ein Objekt (z. B. wenn die Aktion „Ontologie erstellen" durchgeführt wurde, dann liegt das Artefakt „Ontologie" anschließend vor). Sowohl Objektfluss als auch Kontrollfluss können gleichzeitig in dasselbe Objekt eingehen. Bei einer solchen Überlagerung kann auf die Darstellung des Kontrollflusses verzichtet werden.[692]

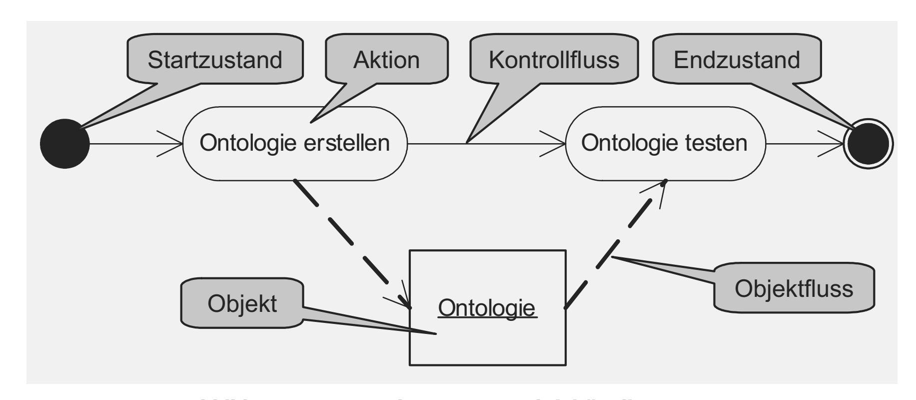


Abbildung 34: Hauptelemente von Aktivitätsdiagrammen

690) In den vorherigen Versionen der UML (kleiner Version 2.0) wurden die Verbindungen zwischen den Aktionen als Transitionen bezeichnet. Dies resultiert aus der Tradition, ein Aktivitätsdiagramm als Sonderform eines Zustandsdiagramms aufzufassen. Transitionen wurden dort dann auch als Zustandsübergänge (ohne Auslöser) bezeichnet (vgl. Booch, Rumbaugh et al. (1999), S. 297). In der Version 2.0 werden Transitionen nur noch für Automaten (state machines) berücksichtigt (vgl. Rumbaugh, Jacobson et al. (2005), S. 657).
Die Begriffe *Fluss* und *Kante* werden sowohl in der UML-Spezifikation als auch hier synonym verwendet (vgl. Fowler (2004), S. 124).

691) Vgl. Oesterreich (2002), S. 91.

692) Um das implizite Wissen nicht unnötig zu erhöhen und die Nachvollziehbarkeit zu erleichtern, wird im OntoFMEA-Vorgehensmodell hierauf verzichtet, d. h. alle Kanten werden explizit dargestellt.

Verzweigung: Mit einer *Verzweigung* wird das weitere Vorgehen im Modell an Bedingungen geknüpft (in UML auch als Wächterbedingungen bezeichnet). Mittels einer Raute werden Bedingungen grafisch repräsentiert. Die Wächterbedingungen, die jeweils erfüllt sein müssen, damit die nächste Aktion durchgeführt werden kann, werden in eckigen Klammern an die ausgehenden Kanten gehängt. Die Abbildung 35 verdeutlicht diesen Umstand beispielhaft. Eine Verzweigung kann auch mehr als zwei Möglichkeiten beinhalten, jedoch müssen sich diese Möglichkeiten disjunkt zueinander verhalten (exklusives oder).

Zusammenführung: Eine Raute, in die mehrere Kanten münden, wird als *Zusammenführung* bezeichnet. Eine ausgehende Kante, die aus der Zusammenführung herausgeht, wird ausgelöst, sobald eine der eingehenden Transitionen ausgelöst wurde.

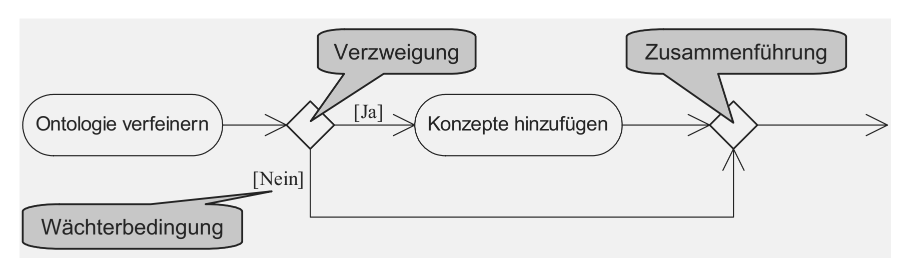


Abbildung 35: Verzweigung und Zusammenführung in Aktivitätsdiagrammen

Aufspaltung: Mittels einer *Aufspaltung* wird eine einzelne Kante in mehrere Kanten aufgeteilt, die alle sofort zur jeweils modellierten Aktion übergehen (siehe Abbildung 36), d. h. es werden parallel Aktionen durchgeführt. Während bei einer Verzweigung lediglich eine Kante weiter feuert, feuern bei einer Aufspaltung mehrere Kanten gleichzeitig. Dies ermöglicht die Darstellung von nebenläufigen Prozessen. Grafisch wird die Aufspaltung durch einen Balken dargestellt. In den Balken geht jeweils eine Kante hinein und mehrere Kanten gehen heraus.

Synchronisation: Mittels einer *Synchronisation* werden nebenläufige Aktionen zusammengeführt, so dass anschließend eine einzelne Aktion weiterverfolgt wird. In eine Synchronisation gehen mehrere Kanten ein und nur eine Kante geht heraus. Die Synchronisation wird grafisch ebenfalls mittels eines Balkens dargestellt.

Konnektor: Weil die zusammenhängende Darstellung des gesamten Modells (bspw. auf einer Druckseite) oft nicht möglich ist, werden *Konnektoren* für die Modelldarstellung eingesetzt, um Modellausschnitte eindeutig (bspw. auf jeweils einer Druckseite) darstellen zu können. Ein Konnektor wird dabei jeweils an die Kante, die zu einem Teil des Modells führt, der nicht mehr abgebildet werden konnte, als Platzhalter gesetzt. Die einzelnen Konnektoren werden über Kreise grafisch dargestellt und besitzen jeweils eine eindeutige Klassifizierung in Form von Buchstaben. Sie werden immer paarweise eingesetzt.

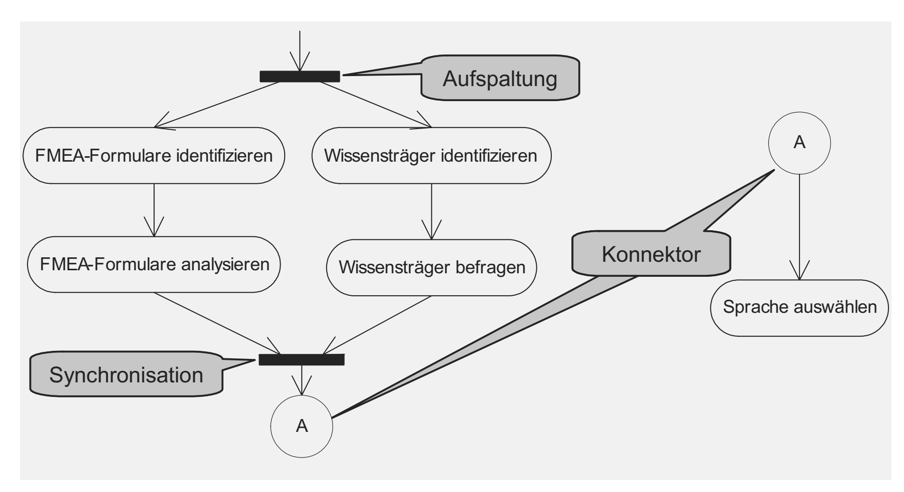


Abbildung 36: Aufspaltung, Synchronisation und Konnektor im Aktivitätsdiagramm

5.3 Explikation von Wissen in Vorgehensmodellen

Ähnlich zu den beiden vorangestellten Instrumenten FMEA und Ontologien werden die vorgestellten Vorgehensmodelle eingehender betrachtet. Im Sinne eines Explikationsvorgangs werden einige wichtige Punkte aufgezeigt, die bei der Anwendung der entsprechenden Vorgehensmodelle implizit enthalten sind. Diese Punkte sollen auf die späteren Untersuchungen vorbereiten. Dabei wird insbesondere Wert auf die Analyse der Vorgehensmodelle des Ontology Engineering gelegt, weil sich hiervon die meisten Erkenntnisse versprochen werden für das Vorgehensmodell OntoFMEA, bei dem sich der Fokus ebenfalls auf die Entwicklung einer Ontologie richtet.

5.3.1 Wissen in Vorgehensmodellen des Software Engineering

Ausgewählte Ansätze aus dem Software Engineering wurden vorgestellt, und es wurde dabei auf ihre Besonderheiten eingegangen. Im Folgenden werden die vorgestellten Ansätze des Software Engineering hinsichtlich ihrer Vor- und Nachteile bei der Anwendung dieser Ansätze zur Gestaltung des Wissens in Wissensbasierten Systemen kurz diskutiert.

Der *sequentielle Software-Life-Cycle-Ansatz* bildet eine wichtige Grundlage für die ingenieurmäßige Entwicklung von Software. Der Vorteil dieses Ansatzes liegt in seiner übersichtlichen Struktur, in der die wichtigsten Aktivitäten des Entwicklungsprozesses definiert und gegeneinander abgegrenzt werden. Dem stehen jedoch mehrere gravierende Nachteile gegenüber:

- Der sequentielle Software-Life-Cycle-Ansatz basiert auf der Prämisse, dass die Anforderungen an die Software nach Abschluss der Spezifikationsphase vollständig erfasst sind. Angesichts der Komplexität der zu erstellenden Software ist eine vollstän-

dige Erfassung der Anforderungen durch die Anwender zu diesem Zeitpunkt oft unmöglich. Änderungs- und Erweiterungswünsche ergeben sich erst mit dem wachsenden Verständnis für den Anwendungsbereich während des Softwareentwicklungsprozesses.[693]

- Die strikte Trennung der einzelnen Phasen stellt zudem häufig eine ungerechtfertigte Idealisierung dar. Denn in der Praxis lassen sich vielfältige Überlappungen der einzelnen Phasen feststellen, und die Interdependenzen sind komplexer, als es in dem sequentiellen Software-Life-Cycle-Ansatz dargestellt wird.

- Die sequentielle Vorgehensweise hat zur Konsequenz, dass erst spät „greifbare" Ergebnisse erstellt werden, die unter realen Bedingungen evaluiert werden können. Software-Fehler oder Änderungswünsche der Auftraggeber werden erst sehr spät identifiziert und können nur unter hohen Kosten berücksichtigt werden.

Der *Wasserfall-Ansatz* hat gegenüber dem sequentiellen Software-Life-Cycle-Ansatz zum einen den Vorteil, dass am Ende einer jeden Phase eine Validierung vorgesehen ist. Hierdurch können die Phasenergebnisse hinsichtlich ihrer Korrektheit und Vollständigkeit überprüft und bei Bedarf angepasst werden. Zum anderen sind im Wasserfall-Ansatz Rückkopplungen zu voranstehenden Phasen vorgesehen, so dass nachträglich neue Erkenntnisse in vorangegangene Phasen(ergebnisse) integriert werden können.[694]

Wie auch am sequentiellen Software-Life-Cycle-Ansatz lässt sich am Wasserfall-Ansatz kritisieren, dass die Spezifikation der Anforderungen nur in der Entwurfsphase vorgesehen ist, abgesehen von der Rückkopplungsmöglichkeit aus der Analysephase. Die späteren Benutzer werden nicht weiter in den Softwareentwicklungsprozess involviert.

Das *prototypbasierte* Vorgehen bietet gegenüber den zuvor vorgestellten Ansätzen den Vorteil, dass gerade eine frühzeitige Entwicklung der Software oder der Softwarekomponenten beabsichtigt wird. Hierdurch kann schon frühzeitig der Prototyp unter realen Bedingungen hinsichtlich der gewünschten Anforderungen der Anwender getestet und bei Bedarf modifiziert werden. Jedoch kann die wiederholte Entwicklung von Prototypen für eine Systemkomponente insbesondere bei großen Software-Projekten erhebliche Kosten verursachen.

Im *Spiral-Ansatz* wird versucht, die Stärken der anderen Ansätze durch Integration oder Behandlung als Sonderfälle zu berücksichtigen. Die Vorteile des Spiral-Ansatzes liegen in der expliziten Berücksichtigung des Risikos innerhalb des Softwareentwicklungsprozesses. So können schon frühzeitig bei der Entwicklung von Prototypen ungeeignete Lösungsvarianten identifiziert und Software-Fehler beseitigt werden. Aufgrund der erhöhten „Komplexitätskosten", die bspw. durch die wiederholte und oft zeitintensive Risikoanalyse entstehen, eignet sich der Ansatz nur für „große" Entwicklungsprojekte.

693) Vgl. Biethahn, Muksch et al. (2000), S. 202.

694) Abgesehen von der ersten Phase *System-Anforderungen*, für die mangels Objekt keine Rückkopplung vorgesehen wurde.

In der folgenden Tabelle 4 finden sich die vorgestellten Ansätze des Software Engineering einander synoptisch gegenübergestellt:

Art des Vorgehens-modells	**Entwicklungs-prozess**	**Vorteile**	**Nachteile**
Sequentieller Software-Life-Cycle-Ansatz	sequentiell	• Zerlegung des Entwicklungs-prozesses in zeitlich getrennte Abschnitte. • Planung und Überwachung werden leichter kontrollierbar. • Komplexitäts-bewältigung durch Aufgaben-teilung.	• Softwareentwicklung ist kein sequentieller Prozess, sondern jede einzelne Phase kann zu Rückkopplungen auf frühere Phasen führen. Das sequentielle Vorgehen spiegelt nicht die Gegebenheiten der Software-entwicklung wider. • Es wird von einer korrekten und abgeschlossenen Anforderungsspezifikation ausgegangen. Hierbei wird nicht berücksichtigt, dass es zu Missverständnissen zwischen Auftraggeber und -nehmer kommen kann, die nachträglich revidiert werden müssen. • Da eine lauffähige Version erst am Ende des Entwicklungs-prozesses vorliegt, wird die Beseitigung von Fehlern aufwendig und teuer.
Wasserfall-Ansatz	sequentiell mit Rückkopplungen aufeinander folgender Phasen	• Validierung der Phasenergebnisse. • Rückkopplung zu vorhergehenden Phasen bei Vorliegen von neuen Erkenntnissen und Identifikation von Fehlern.	• Siehe Nachteile des sequentiellen Software-Life-Cycle-Modells mit der Einschränkung, dass für aufeinander folgende Phasen eine Rückkopplung vorgesehen ist. Jedoch ist eine Rückkopplung nicht für weiter auseinander liegende Phasen vorgesehen.
Prototyp-basierter Software-Life-Cycle-Ansatz	iterativ	• Frühe Implementierung eines vorläufigen Prototyps, so dass Änderungen und Fehler identifiziert und angepasst bzw. behoben werden können.	• Keine theoretische Systematik, die den Entwicklungsprozess mit Erfahrungswissen unterstützt.
Spiral-Ansatz	iterativ	• Validierung der Phasenergebnisse. • Berücksichtigung des Risikos des Software-entwicklungs-prozesses.	• Eignet sich nur zur Durchführung sehr komplexer, großer Entwicklungsprojekte. • Anwender müssen Verfahren der Risikoanalyse kennen.

Tabelle 4: Synopse Ansätze des Software Engineering

5.3.2 Wissen in Vorgehensmodellen des Knowledge Engineering

Die Abbildung 37 stellt den Zusammenhang zwischen Prototyp-Ansatz und Modellbasiertem Ansatz schematisch dar. In der Abbildung werden die beiden Prozess-Ansätze vereint. Es handelt sich bei der Abbildung um eine erweiterte Sichtweise der Abbildung 25 und der Abbildung 26 unter Berücksichtigung der Ausführungen in Kapitel 5.1.1.4, S. 122. Wie in den Ausführungen erwähnt, lässt sich das Knowledge Engineering in die Phase der *Wissensakquisition* und der *Wissensverwendung* einteilen. Innerhalb der Wissensakquisition finden sich die Phasen der *Wissensidentifikation* und der *Wissensoperationalisierung*. Die Identifikation des Wissens gliedert sich in die *Wissenserhebung* und die *Wissensinterpretation*. Die Wissensoperationalisierung besteht aus der *Konzeptualisierung*, gegebenenfalls der *Modellierung* (mit dem Ergebnis eines explizit vorliegenden Modells) sowie der *Formalisierung* des identifizierten Wissens. Die Wissensverwendung besteht in der Hauptsache aus der *Implementierung* und dem *Test* des (prototypischen) Wissensbasierten Systems.

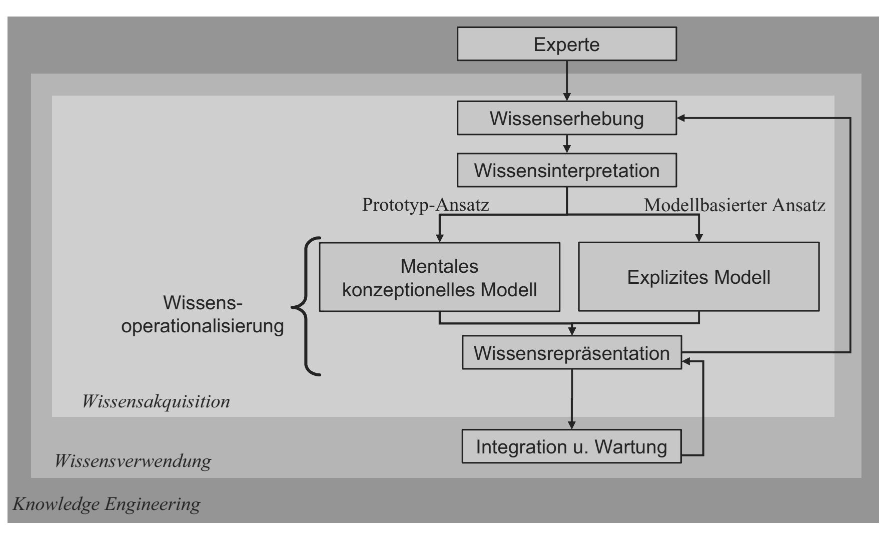


Abbildung 37: Prototyp- vs. Modellbasierter Ansatz im Knowledge Engineering[695]

Der Schwerpunkt des Knowledge Engineering liegt, wie aus der Abbildung ersichtlich wird, insbesondere in der Wissensakquisition. Folgt man der Argumentation, dass jede Formalisierung von Wissen einer vorherigen Konzeptualisierung zumindest mittels eines mentalen Modells bedarf, wird beim Prototyp-Ansatz ein konzeptuelles Modell nur mental repräsentiert, weil die Formalisierung unmittelbar im Anschluss an die Konzeptualisierung erfolgt. Es wird kein explizites Modell erstellt, vielmehr findet sich das Wissen in den Köpfen der Entwickler des Wissensbasierten Systems. Im Gegensatz dazu berücksichtigt der Modellbasierte Ansatz

695) In Anlehnung an Haun (2000), S. 197.

ein explizites Modell in der Phase der Wissensakquisition.[696] Dieses explizite Modell wurde jedoch vorher „in den Köpfen“ der Entwickler mental konzeptualisiert.

Der Ablauf der Ansätze für Vorgehensmodelle des Knowledge Engineering wirkt auf die Ansätze für Vorgehensmodelle des Ontology Engineering, weil das Ontology Engineering als Teilmenge des Knowledge Engineering angesehen wird.[697]

5.3.3 Wissen in Vorgehensmodellen des Ontology Engineering

5.3.3.1 Einführung

Um die Vorgehensmodelle aus dem Bereich des Ontology Engineering insbesondere hinsichtlich ihrer Verwertbarkeit für das zu erstellende OntoFMEA-Vorgehensmodell zu untersuchen, werden im Folgenden zuerst Anforderungen aufgestellt, anschließend werden bekannte Vorgehensmodelle vorgestellt und anhand der Anforderungen analysiert. Bei den vorzustellenden Vorgehensmodellen wird deutlich, dass diese die Besonderheiten bei der Entwicklung von Wissensbasierten Systemen berücksichtigen. Insbesondere stützen sich sämtliche Ontology-Engineering-Vorgehensmodelle auf den Modellbasierten Ansatz, d. h., es wird immer die Spezifikation des zu repräsentierenden Wissens über eine Domäne (inklusive etwaiger Regeln) expliziert. Das repräsentierte Wissen kann in diesem Fall die Form eines (allgemeinen) Modells der Domäne annehmen. Gerade die konstituierende Eigenschaft von Ontologien erfordert, dass eine explizite gemeinschaftliche Repräsentation der zu erstellenden Domäne konzeptualisiert und spezifiziert wird, so dass ein Prototyp-Ansatz im vorgenannten Sinne nicht anwendbar ist.[698] Denn würde davon ausgegangen, dass mittels des Prototyp-Ansatzes ein mentales konzeptuelles Modell unmittelbar implementiert wird, dann wäre die Ontologie als implementierter Prototyp des allgemeinen Wissens über die Domäne nicht das Ergebnis eines geteilten Konzeptualisierungsprozesses. Diese „Gemeinschaftlichkeit“ wird jedoch als konstituierendes Merkmal für Ontologien angesehen.

696) Um die Sonderstellung des expliziten Modells, d.h. der Phase der Modellierung innerhalb des Modellbasierten Ansatzes, hervorzuheben, wurde dieser Teil des Vorgehens in der Abbildung 37 mit einer gestrichelten Umrandung hervorgehoben.

697) Siehe hierzu auch die Ausführungen in Kapitel 2.1.2.5, S. 31 ff.

698) Insofern sollte an dieser Stelle die bisherige Argumentationsführung deutlich werden: Um das avisierte Vorgehensmodell zu entwickeln, werden nacheinander die Zusammenhänge *a maximis ad minima* dargelegt, der Weg führt nämlich vom Software Engineering über das Knowledge Engineering zum Ontology Engineering. Weil insbesondere die Vorgehensmodelle des Ontology Engineering für die Eigenentwicklung von großer Bedeutung sind, werden diese einer genaueren Analyse unterzogen, um die jeweiligen Vor- und Nachteile herauszuarbeiten.

5.3.3.2 Anforderungen an ein Vorgehensmodell

5.3.3.2.1 Allgemeine Grundlagen

Die wichtigsten Anforderungen an das Vorgehensmodell, das in diesem Kontext eingesetzt werden soll, um die Entwicklung von FMEA-Ontologien zu beschreiben und zu strukturieren, werden im folgenden Unterkapitel dargestellt. Im Einzelnen sind dies:

- Generizität
- Anwendungsbezogenheit
- Vollständigkeit
- Dokumentation
- Einfachheit
- Klarheit
- Werkzeugunterstützung

Ähnliche Zusammenstellungen von Anforderungen für die Konstruktion von (Vorgehens-)-Modellen und für ihre Bewertung finden sich bspw. in Verlage (1998a) und in Moody, Shanks (1994).[699] Die hier genannten Anforderungen wurden aufgrund ihrer Bedeutung in der Literatur oder wegen ihrer Relevanz für die vorliegenden spezifischen Gegebenheiten (wie z. B. Praxisrelevanz, spezielle überlappende betriebswirtschaftliche und technische Domäne) ausgewählt.

Bei der Umsetzung der nachfolgend geschilderten Anforderungen ist zu beachten, dass diese nicht unabhängig voneinander bestehen, sondern sich gegenseitig in ihrer Wirkung beeinflussen.

Einerseits verhalten sich einige der Anforderungen komplementär zueinander; so führt zum Beispiel die (möglichst) vollständige Spezifikation eines Entwicklungsprozesses mit einer detaillierten Beschreibung der Phasenaktivitäten und Methoden in den meisten Fällen zu einer hohen Anwendungsbezogenheit. Auch die Einbeziehung von Möglichkeiten zur Prozessunterstützung durch geeignete Computerprogramme (Werkzeugunterstützung) kann zu einer einfacheren Realisierung und damit zur Anwendungsbezogenheit eines Vorgehensmodells beitragen.

Andererseits können Anforderungen auch gegensätzliche Zielsetzungen verfolgen, so dass eine gleichzeitige Umsetzung schwierig ist. Wenn die Klarheit des Vorgehensmodells durch eine formalsprachliche Beschreibung, wie z. B. mittels mathematischer Formeln, realisiert wird, kann sich dies nachteilig auf die Einfachheit und damit auf die Nachvollziehbarkeit des An-

699) Vgl. Verlage (1998a), S.73, und Moody, Shanks (1994), S. 101 f.

satzes auswirken.[700] Oft lassen sich solche Zielkonflikte durch Hinzunahme von Hilfsmitteln auflösen, indem etwa verschiedene Abstraktionsniveaus sowohl eine generelle als auch eine anwendungsbezogene Sicht auf den Entwicklungsprozess ermöglichen oder für die Realisierung der Klarheit und Eindeutigkeit des Vorgehensmodells formale oder semi-formale Modelle, die zusätzlich zur textuellen Beschreibung vorliegen, genutzt werden.

5.3.3.2.2 Generizität

Das Vorgehensmodell soll insofern allgemeine Gültigkeit (Generizität) besitzen, als es nicht ausschließlich für ein bestimmtes Projekt konzipiert ist, sondern auch im Rahmen ähnlicher Vorhaben, etwa grundsätzlich für die Entwicklung ontologiebasierter Systeme, zum Einsatz kommen kann. Ein generelles, auch domänenunabhängiges Vorgehensmodell ist eine Grundvoraussetzung für die allgemeine Anerkennung und die Reife der Disziplin der Ontologieentwicklung.[701] Dadurch wird einerseits eine Wiederverwendung des Wissens über das Vorgehen bei der Entwicklung von Ontologien ermöglicht, andererseits kann ein Entwicklungsprozess leichter projektspezifisch angepasst werden, weil die Möglichkeit der Anpassung von Anfang an als eine Bedingung bei der Entwicklung eines Vorgehensmodells berücksichtigt wurde.[702] Dabei ist zu beachten, dass derzeit kein spezifisches Vorgehensmodell zur Entwicklung von FMEA-Ontologien vorliegt, so dass der Verfasser sich am ehesten von bereits existierenden generischen Vorgehensmodellen Erkenntnisse für die Entwicklung eines Vorgehensmodells zur Konstruktion von FMEA-Ontologien erhofft.

5.3.3.2.3 Anwendungsbezogenheit

Damit das Vorgehensmodell im Falle seines Einsatzes eine Hilfestellung zur Umsetzung eines Softwareentwicklungsprojekts leisten kann, muss es auch konkrete Beispiele und Handlungsempfehlungen für den Anwender des Vorgehensmodells umfassen. Hierzu zählt zum Beispiel eine exemplarische Detaillierung von Aktivitäten, um das methodische Fundament eines Ansatzes zu erläutern.[703] Diese Anforderung steht nicht zwangsläufig im Gegensatz zum Kriterium der Generizität; bspw. könnte eine Anwendungsbezogenheit dadurch realisiert werden, dass zusätzlich zu einer allgemeinen Phasenbeschreibung jeweils Beispiele mögliche Umsetzungen verdeutlichen.

700) Hierüber lässt sich jedoch vortrefflich streiten, weil auch der Standpunkt vertreten werden kann, dass einem geübten Anwender eine formalsprachliche Beschreibung gerade die Nachvollziehbarkeit erleichtert.
Ein weiterer Zielkonflikt lässt sich zwischen den Anforderungen nach allgemeiner Gültigkeit und Vollständigkeit eines Vorgehensmodells möglicherweise ausmachen. Dabei wird unterstellt, dass eine allgemeine Gültigkeit sich oftmals nur unter Verzicht einer vollständigen Darstellung aller möglichen Aspekte eines Vorgehens erreichen lässt. Je detaillierter spezifische Aspekte eines bestimmten Vorgehens dargelegt werden, desto weniger Allgemeingültigkeit lässt sich im Allgemeinen für eine solche Darlegung reklamieren.

701) Vgl. Guarino, welty (2002), S. 61.

702) Auch spezifische Vorgehensmodelle können wiederverwendet werden, wenn die entsprechenden Rahmenbedingungen erfüllt sind. Nur wird die Wahrscheinlichkeit der Wiederverwendung deutlich geringer eingeschätzt, nicht zuletzt, weil diese Wiederverwendung ursprünglich auch nicht beabsichtigt war.

703) Vgl. Fernández López (1999), S. 4.3.

5.3.3.2.4 Vollständigkeit

Das Vorgehensmodell soll im Sinne der Anwendungsziele[704] vollständig sein. Es sollen von der Ausgangssituation bis zur Zielerreichung alle relevanten Aspekte dargestellt werden. Das bedeutet auf der einen Seite, dass aus Sicht der Benutzer alle notwendigen und sinnvollen Schritte enthalten sein müssen. Auf der anderen Seite ist eine ausreichend genaue Beschreibung der einzelnen Schritte erforderlich, damit keine „Lücken" entstehen, die bei der konkreten Projektdurchführung noch geschlossen werden müssen. Diese relative Definition von Vollständigkeit[705] führt dazu, dass die Erfüllung dieser Anforderung bei vielen Ansätzen im Vorhinein nicht überprüfbar ist, da die Anwendungsziele, Annahmen (im Sinne von Prämissen hinsichtlich des Anwendungskontexts) und Ausgangssituationen oft nur implizit vorhanden oder nicht detailliert genug beschrieben sind.[706]

5.3.3.2.5 Dokumentation

In der Literatur zu Vorgehensmodellen herrscht in den meisten Fällen Übereinstimmung darin, dass eine umfassende und stetige Dokumentation für Projekte jeder Art von essentieller Bedeutung ist und eine Selbstverständlichkeit sein sollte.[707] Dennoch erscheint es sinnvoll, diese Aktivität als Bestandteil und Anforderung an ein Vorgehensmodell explizit zu fordern, weil gerade die auf der Hand liegenden Aktivitäten oftmals Gefahr laufen, vernachlässigt zu werden.

Während jeder Phase sollen die ursprünglichen Intentionen und die Design Rationale[708] natürlichsprachlich (und nicht formalsprachlich) ausgeführt werden. Dieses Vorgehen erleichtert später eventuelle Rechtfertigungen gegenüber Änderungsvorschlägen. Außerdem kann durch regelmäßige Dokumentation die Nachvollziehbarkeit der Ontologieentwicklung und somit auch die Akzeptanz unter Benutzern und Entwicklern bedeutend verbessert werden. Ein wei-

704) Als Anwendungsziele des Vorgehensmodells werden die konkretisierten Mittel, die der Benutzer des Vorgehensmodells im Vorhinein festlegt und die zur erfolgreichen Ontologieentwicklung und zur Implementierung der Ontologien in ein lauffähiges System dienen, bezeichnet.

705) Vgl. z. B. Schütte (1997), S. 7. Dort wird ein Modell als vollständig angesehen, wenn es alle im Meta-Modell beschriebenen Beziehungen zwischen den Objekten in der gleichen Form einhält. Moody und Shanks (1994), S. 102, betrachten ein (Daten-) Modell als vollständig, sobald es die Fähigkeit besitzt, alle funktionalen Anforderungen der Benutzer umzusetzen. Diese Definitionen von Vollständigkeit richten sich also ebenfalls nicht nach einem *allgemeingültigen* Referenzmodell, sondern sind abhängig von dem vorgegebenen Meta-Modell oder von den Benutzern des Modells.

706) Aus diesem Grund wird das Kriterium der Vollständigkeit an dieser Stelle zwar aufgeführt und erläutert, jedoch nicht zur Beurteilung der dargestellten Ansätze eingesetzt. Zwar wäre, wie bereits erwähnt, eine Bewertung der Vollständigkeit aus Sicht der Anwender des Vorgehensmodells (also der Ontologieentwickler) möglich (bspw. mittels einer Referenzanwendung). Die Arbeiten hierzu würden aufgrund ihres Umfangs jedoch zu weit vom Fokus dieser Arbeit abrücken. Darüber hinaus versprechen die zu erwartenden Ergebnisse keinen weiteren nicht-trivialen Mehrwert für den Argumentationsverlauf dieser Arbeit.

707) Siehe Kapitel 5.3.3.3.3, S. 177.

708) Als Design Rationale wird hier die Zusammenfassung von Merkmalen einer Gestaltungsentscheidung und deren Begründungen verstanden. Vgl. zum Begriff „Design Rationale" Moran, Carrol (1996), S. 1 ff.

terer Vorteil ist, dass solche zusätzlichen Informationen nachfolgende Änderungen für konkrete Anwendungen vereinfachen. Aus diesen Gründen soll das Vorgehensmodell die Dokumentation des Entwicklungsprozesses berücksichtigen und konkrete Methoden zur Realisierung der Dokumentation nennen.

5.3.3.2.6 Einfachheit

Simplizität, bezogen auf die Größe und Komplexität eines Modells[709], ist wünschenswert, weil der Inhalt des Vorgehensmodells für alle Benutzer einfach nachvollziehbar sein soll. Insbesondere bei der Entwicklung von wirtschaftswissenschaftlichen Software-Applikationen, die gegebenenfalls den Weg in die Praxis finden sollen, ist es wichtig, dass das Vorgehensmodell nicht zu komplex aufgebaut ist, weil während eines großen Teils der Projektdurchführung „Nicht-Software-Techniker“ beteiligt sind. Simplizität ist deshalb für eine hohe Akzeptanz und auch Effizienz des Vorgehensmodells von großem Vorteil.[710]

5.3.3.2.7 Klarheit

Der Grundsatz der Klarheit bezieht sich auf „...die Verständlichkeit und die Eindeutigkeit von Modellsystemen“[711]. Das Vorgehensmodell soll verständlich und eindeutig formuliert sein, um unterschiedliche Interpretationen und eventuell zeitaufwendige Diskussionen unter den Projektverantwortlichen zu vermeiden. Die Anforderung ist wichtig für Anwendbarkeit und Akzeptanz des Vorgehensmodells, denn alle Modelle, insbesondere Vorgehensmodelle, dienen zur Veranschaulichung und damit immer auch als Diskussionsgrundlage, so dass Zweideutigkeiten nicht nur zu Missverständnissen führen, sondern auch die Qualität des Entwicklungsprozesses und der resultierenden Ontologie negativ beeinflussen können. Aus diesem Grund ist die Klarheit auch abhängig von der *Konsistenz*, also der syntaktischen Korrektheit, sowie der *Widerspruchsfreiheit* eines Modells. Konsistenz und Widerspruchsfreiheit werden

---

709) Eine ähnliche Erläuterung geben Moody, Shanks (1994), S. 102. Für die Messung der Komplexität eines Datenmodells schlagen sie vor, die Anzahl der Entitäten und die Menge der Beziehungen im jeweiligen Datenmodell zu addieren. Für Vorgehensmodelle könnten die Anzahl der Phasen und der Detailgrad der Phasenbeschreibungen sowie zusätzlich die Anzahl von Rückkopplungen, von Verzweigungen und von parallel ausführbaren Schritten als Maßstäbe für die Simplizität herangezogen werden.

710) Die geforderte Einfachheit bezieht sich somit vornehmlich auf die Reduktion der Komplexität eines Modells, bspw. sollte die *Nebenläufigkeit* nur dann, wenn es unbedingt notwendig erscheint, in einem Modell berücksichtigt werden. Die Nebenläufigkeit birgt in der Anwendung die Gefahr, dass Zuständigkeiten und Termine abgewälzt bzw. verschoben werden, weil davon ausgegangen wird, dass jeweils der andere „Strang“ zuständig bzw. verantwortlich für die Terminhaltung ist.
Eine *effiziente Anwendung* des Vorgehensmodells wird erreicht, wenn alle Benutzer das Vorgehensmodell akzeptieren und so anwenden, dass möglichst Ressourcen schonend die angestrebten Ergebnisse erreicht werden. Vermeiden von Nebenläufigkeit und effiziente Anwendung sind somit Bestandteile des Kriteriums „Einfachheit“.

711) Schütte (1997), S. 10.

als Unterkriterien der Klarheit berücksichtigt. Das Kriterium der Klarheit kann realisiert werden, indem die Modellelemente möglichst formalsprachlich beschrieben werden.[712]

5.3.3.2.8 Werkzeugunterstützung

Das Vorgehensmodell soll für den beschriebenen Entwicklungsprozess und die dafür vorgeschlagenen Methoden entsprechende IT-Werkzeuge empfehlen, die die einzelnen Aktivitäten unterstützen und die Qualität der Ergebnisse verbessern können. Die Umsetzung dieser Anforderung trägt ebenfalls zu einer leichteren Anwendung des Vorgehensmodells und einer höheren Akzeptanz unter den Beteiligten bei, da der Komfort und auch die Effizienz des Entwicklungsprozesses durch geeignete Werkzeugunterstützung bedeutend gesteigert werden können. Der Einsatz einer Software als Werkzeug zur kollaborativen Textbearbeitung etwa vereinfacht das gemeinsame Arbeiten mehrerer Personen an einem Dokument.

5.3.3.3 Evaluation der Ansätze des Ontology Engineering

Die in den vorigen Kapiteln dargestellten Vorgehensmodelle setzen verschiedene Schwerpunkte und lassen sich für einzelne Entwicklungsprojekte unterschiedlich einsetzen. Um eine Bewertung der Ansätze hinsichtlich ihrer Eignung für die Konstruktion von FMEA-Ontologien vornehmen zu können, werden im Folgenden die in Kapitel 5.3.3.2 aufgestellten Anforderungen zur Evaluation herangezogen. Für jede Anforderung wird untersucht, ob und inwieweit sie in den einzelnen Ansätzen erfüllt wird. Anschließend werden die Ergebnisse dieser Analyse, also der Grad der Erfüllung der jeweiligen Anforderung durch die verschiedenen Vorgehensmodelle, in einem gesonderten Kapitel in einer Tabelle zusammengefasst.

5.3.3.3.1 Generizität

Der IDEF5-Ansatz ist im Kern sehr generisch, weil er im Vorhinein keinerlei Einschränkungen bezüglich der Domäne vorsieht. Allerdings verfügt der Ansatz mit den beiden vorgegebenen Sprachen über eine Einschränkung, die die Generizität beeinträchtigt. Zwar basiert die *Elaboration Language* auf KIF und ist damit prinzipiell austauschbar gegenüber weiteren an KIF angelehnten Sprachen, jedoch werden damit Repräsentationssprachen, die für bestimmte Anwendungsfälle prädestiniert erscheinen würden, weil sie bspw. die Quantifizierung über Prädikate oder nichtmonotones Schließen erlauben, ausgeschlossen.[713]

Dem Kriterium der Generizität wird in dem von Uschold und King vorgestellten Enterprise-Model-Ansatz besondere Aufmerksamkeit geschenkt. Bevor die Autoren detailliert auf ihre

712) Ein besonders hoher Grad an Formalität geht allerdings oft zu Lasten der Nachvollziehbarkeit eines Vorgehensmodells. An dieser Stelle kann die Kombination informaler Ausführungen mit (semi-) formalen Beschreibungen wie Modellen dazu beitragen, dass gleichzeitig die Simplizität und Nachvollziehbarkeit sowie die Klarheit eines Vorgehensmodells gewährleistet sind.
Bspw. wäre eine (semi-) formale Vorgehensmodellierung mittels Ereignisgesteuerter Prozessketten (EPK) möglich. Zum Begriff der *Ereignisgesteuerten Prozessketten* vgl. Scheer (1999). Für eine weitere alternative Darstellungsform siehe Kapitel 5.2, S. 162 ff.

713) Aus diesem Grund werden auch Anwendungen für das Semantic Web nur mittelbar ermöglicht.

eigenen Erfahrungen mit der Ontologieentwicklung eingehen, beschreiben sie ein allgemeines, grob formuliertes Vorgehensmodell[714], das auf ein sehr breites Spektrum von Projekten zur Ontologieentwicklung angewendet werden kann.

Der TOVE-Ansatz von Grüninger und Fox ist weniger generisch konzipiert. Zwar lässt sich auch dieses Modell auf viele Ontologieentwicklungsvorhaben übertragen, die Art der Durchführung ist aber zum Teil schon durch die Strukturierung der Phasen vorgegeben. So wird zum Beispiel bei TOVE die Spezifikation der Anforderungen durch Kompetenzfragen realisiert. Im Prinzip lassen sich Kompetenzfragen immer stellen, jedoch müssen bei diesem Vorgehen sämtliche Anforderungen an die Ontologie bereits im Vorfeld bekannt sein. Für bestimmte Domänen, die bspw. zuvor überhaupt noch nicht konzeptualisiert wurden, muss ein Konsens bei den Akteuren, die an der Ontologieentwicklung beteiligt sind, hinsichtlich der enthaltenen Konzepte, Relationen und Regeln erst noch gemeinschaftlich ermittelt werden. Hier wird deutlich, dass zu Beginn der Entwicklung einer Ontologie zur Repräsentation des Wissens einer Domäne höchstens zufällig sämtliche Anforderungen in Form von Kompetenzfragen, die von allen Akteuren gemeinschaftlich akzeptiert werden und die wie geschlossene Fragen funktionieren, ermittelt werden können.

Die Autoren von METHONTOLOGY lassen dagegen sowohl die Wahl der formalen Sprache als auch die Art der Anforderungsspezifikation offen. Sogar die Reihenfolge der Aktivitäten, deren Durchführung sie empfehlen, kann für jedes Projekt neu entschieden werden. Die Generizität des Ansatzes wird besonders deutlich durch die allgemeine Formulierung der Phasen und durch die große Anzahl an Alternativen, die für ihre Umsetzung vorgeschlagen werden.

Auch der On-To-Knowledge-Ansatz umfasst eine eher grobe Beschreibung der Aktivitäten, die durch Fallstudien detailliert werden. Das Vorgehensmodell ist auf andere Projekte übertragbar, jedoch nur auf solche, bei denen die Ontologie den Kern eines Wissensmanagementsystems bildet und in deren Rahmen die Ontologieentwicklung Teil der Wissensmanagementsystementwicklung ist.

Bei der Arbeit von Holsapple und Joshi liegt der Fokus ausschließlich auf der Ontologieentwicklung (unabhängig von ihrem Umfeld). Der vorgestellte Kollaborative Ansatz erfüllt das Kriterium der Generizität nur dann, wenn bei Projekten zur Ontologieentwicklung grundsätzlich mehrere Personen mit verschiedenen Ansichten beteiligt sind.

5.3.3.3.2 Anwendungsbezogenheit

Die meisten der vorgestellten Ansätze lassen sich als relativ detailliert und anwendungsnah einstufen.

Die Anwendungsbezogenheit wird beim IDEF5-Ansatz durch zahlreiche Formblätter sichergestellt. Diese Formblätter sollen eine nachvollziehbare Anwendung ermöglichen. So gibt es

714) Vgl. Uschold, King (1995), S. 2. Dieses Modell bildet jedoch nur ein Grundgerüst und sollte nach Aussage der Autoren durch entsprechende Erweiterungen sowie durch die Festlegung von Methoden an konkrete Projekte angepasst werden.

Formblätter etwa für die *Zwecke der Ontologieentwicklung*, *Verwaltung des Quellmaterials* und in die Ontologie *aufzunehmende Konzepte*. Der Ansatz verfügt allerdings nicht über konkrete Handlungsempfehlungen für den Benutzer. Nur in Teilen wird der Anspruch – eine „Ontology Capture Method" zu sein – erfüllt.

Der Enterprise-Model-Ansatz hingegen erhebt nur den Anspruch, ein Grundgerüst für das Vorgehen bei der Ontologieentwicklung bereitzustellen. Die einzelnen Phasen werden genannt und erläutert. Eine genaue Beschreibung und Hinweise zur Umsetzung werden jedoch nicht gegeben. Allerdings gilt dies nur für das Vorgehensmodell selbst, denn in ihrem Artikel von 1995 gehen die Autoren anschließend auf ihre Erfahrungen aus einer Fallstudie ein, wobei sie konkrete Methoden zur Umsetzung ihres Vorgehensmodells darstellen.[715] Teilweise wird dadurch eine Erweiterung und Verfeinerung des Modells gegeben, doch für viele Vorgehensschritte nutzen Uschold und King die Vorschläge und Erkenntnisse aus anderen Arbeiten und verweisen lediglich auf die jeweiligen Literaturquellen.

Bei TOVE ist stattdessen die Beschreibung der Implementierungsmöglichkeiten und der Aktivitäten Bestandteil des Vorgehensmodells selbst. So werden bspw. konkrete Empfehlungen bezüglich der Anforderungsspezifizierung (Kompetenzfragen) und der Wahl der formalen Sprache (Prädikatenlogik) gegeben und die Vorgehensschritte bei der Formalisierung relativ detailliert erläutert.

Auch der Ansatz METHONTOLOGY beinhaltet bereits einige Beschreibungen von Methoden für die Durchführung einer Ontologieentwicklung. Obwohl die Autoren den Anspruch erheben, einen hohen Detaillierungsgrad zu erreichen[716], sind die Phasenaktivitäten in den vorliegenden Quellen nur teilweise durch Realisierungshinweise konkretisiert. Dabei ist nicht festgelegt, auf welche Art und Weise das Vorgehensmodell umzusetzen ist, aber es werden für jede Phase mehrere Alternativen zur Realisierung der Aufgaben vorgestellt. So können die Projektverantwortlichen sich zum Beispiel bei der Wissensakquisition für strukturierte oder unstrukturierte Interviews, für Brainstorming, Textanalyse oder für eine computerunterstützte Wissenserhebung entscheiden.

Der On-To-Knowledge-Ansatz kann ebenfalls als anwendungsbezogen angesehen werden. Dabei konzentrieren sich die Autoren besonders darauf, die Ergebnisse der einzelnen Phasen, also etwa das Anforderungsspezifikationsdokument oder das Kompetenzfragen-Formular, genau zu beschreiben und ihren Aufbau durch Beispiel-Grafiken zu verdeutlichen.

Im Kollaborativen Ansatz sind die Phasen des Vorgehensmodells und die Möglichkeiten zu ihrer Umsetzung auch sehr detailliert dargestellt. Damit erfüllt die Arbeit in dieser Hinsicht die Anforderung der Anwendungsbezogenheit. Allerdings liegt der Schwerpunkt dieses Ansatzes auf der Verfeinerung von Ontologien durch mehrere beteiligte Personen, während die Erstellung und Formalisierung einer ersten Basis-Ontologie nicht ausreichend genau behandelt wird, um den Ansatz in einem konkreten Projekt ohne weitere methodische Fundierung

715) Vgl. Uschold, King (1995), S. 4 ff.

716) Vgl. Fernández López (1999), S. 4.9.

anwenden zu können. Der Ansatz ist hauptsächlich für große und komplexe Projekte geeignet, weil der Einsatz der Delphi-Methode im Kontext der Ontologieentwicklung einen hohen Aufwand und umfassende Kenntnisse der Beteiligten über die Domäne sowie über Ontologien erfordert.

5.3.3.3.3 Dokumentation

Der IDEF5-Ansatz verfügt über zahlreiche Dokumentationsvorgaben, die teilweise in ihrer bürokratischen Wirkung hemmend sein können. So gibt es bspw. eine Liste mit den in die Ontologie aufzunehmenden Konzepten (*Term Pool*) und eine Liste mit den Beschreibungen dieser Konzepte (*Term Description Form*), die lediglich zusätzlich eine kurze Beschreibung zum jeweiligen Konzept vorhält. Relationen zu weiteren Konzepten werden hier noch gar nicht erfasst und erst später in einem weiteren Dokument abgelegt.

Uschold und King definieren das Dokumentieren und Begründen von Entscheidungen und Annahmen als eigene Phase ihres Ansatzes und stellen damit die Beachtung dieser wichtigen Aufgabe bei einer Anwendung des Enterprise-Model-Ansatzes sicher. Jedoch ist an dieser Stelle zu erwähnen, dass in den eigentlichen Entwicklungsphasen sowie bei der Darstellung der Fallstudie nicht mehr näher auf diese Tätigkeit eingegangen wird.

In dem Vorgehensmodell von TOVE ist die Dokumentation der relevanten Entscheidungen und Ergebnisse eher implizit enthalten. So wird bspw. durch die Formulierung von Motivations-Szenarien und Kompetenzfragen das Resultat der vorangegangenen Aktivitäten festgehalten, die dabei vorhandenen personen- und situationsabhängigen Annahmen und Einschränkungen werden jedoch in diesem Ansatz nicht beachtet.

Die Autoren von METHONTOLOGY hingegen präsentieren die Aufgabe der Dokumentation einerseits als eigenständige Phase in ihrem Vorgehensmodell und gleichzeitig auch als Bestandteil jeder einzelnen Ontologieentwicklungsphase. Die Dokumentation wird dabei als entscheidende Aktivität während des gesamten Entwicklungsprozesses dargestellt und durch die verschiedenen Artefakte realisiert, die im Laufe des Projekts entstehen, wie zum Beispiel Anforderungsspezifikation, Wissensakquisitionsdokument und konzeptuelles Modell.

Im Gegensatz dazu wird dem Grundsatz der Dokumentation bei der Beschreibung des On-To-Knowledge-Ansatzes wenig Aufmerksamkeit geschenkt. Er findet durch die Struktur des Vorgehensmodells keine Erwähnung, und auch in den wenigen während des Entwicklungsprozesses produzierten Artefakten werden Entscheidungen und ihre Begründungen allenfalls implizit dokumentiert (bspw. wird die Auswahl einer Repräsentationssprache lediglich dadurch deutlich, dass eine bestimmte formalsprachliche Ontologie vorliegt; es wird am Ende nicht mehr deutlich, aus welchen Sprachen die verwendete Sprache ausgewählt wurde).

Der Kollaborative Ansatz von Holsapple und Joshi enthält die Dokumentation auch nicht als eigenen Bestandteil des Phasenmodells, doch in jedem Vorgehensschritt werden die wichtigsten Ergebnisse in Form von schriftlichen Ausführungen als Entscheidungsgrundlage an die nächsten Phasen weitergegeben. So soll etwa beim Einsatz der Expertenbefragung in der Ver-

feinerungsphase in jeder Iteration eine Zusammenfassung der eingegangenen Kritikpunkte und Kommentare angefertigt werden.

5.3.3.3.4 Einfachheit

Da die Einfachheit eines Vorgehensmodell zu einem großen Teil durch die Komplexität seines Aufbaus beeinflusst wird, kann ein so generelles und grob strukturiertes Grundgerüst, wie ihn etwa der TOVE-Ansatz bietet, dieses Kriterium leichter erfüllen als ein umfassenderes, ausführlich beschriebenes Vorgehensmodell.

Die fünf Hauptphasen des IDEF5-Ansatzes lassen sich einfach erkennen und bieten in ihrer Zusammensetzung ein intuitives Vorgehen bei der Ontologieentwicklung. Die Verwendung der *Elaboration Language* erfordert jedoch umfangreiche Fachkenntnisse, die von Personen mit geringen Vorkenntnissen nicht erbracht werden können. Hinzu kommt, dass kein eigentlicher Startpunkt der Ontologieentwicklung angegeben wird, d. h. der Ontologieentwickler beginnt mit einer losen Sammlung von Konzepten, die bei der Datenanalyse (zufällig) vorgefunden werden und die mehr oder weniger zufällig durch Relationen geordnet werden.

Das von Uschold und King entwickelte Vorgehensmodell besteht aus nur vier Phasen, von denen die umfassendste (die Ontologieentwicklung selbst) durch die Unterteilung in drei Vorgehensschritte weiter detailliert wird. Auch die Phasenbezeichnungen und ihre Beschreibung der Aktivitäten sind wenig technisch. Für weitere Einzelheiten wird oft auf andere Quellen verwiesen, so dass das Modell relativ simpel gehalten und für die meisten Personen mit geringen Vorkenntnissen verständlich ist. Jedoch erscheint der Verweis auf andere, im Allgemeinen nicht unmittelbar zugängliche Quellen bei einer tiefergehenden Anwendung als weniger einfach, als es bspw. mit einem vollständig geschlossenen Vorgehensmodell denkbar wäre. Zudem bleibt die Komplexität der externen Quellen unbeachtet.

Beim Einsatz des TOVE-Ansatzes dagegen ist spätestens bei der formalen Spezifikation der Kompetenzfragen und anschließend der Axiome und der Vollständigkeitstheoreme logisch-mathematisches Vorstellungsvermögen erforderlich, um die einzelnen Schritte sowie die Erläuterungen und Definitionen nachvollziehen zu können. Natürlich wird die tatsächliche Formalisierung meist von Personen mit Informatik- oder Mathematik-Kenntnissen vorgenommen, doch es könnte bei TOVE schon bei der Planung zu Verständnis- oder sogar Akzeptanzschwierigkeiten auf der Seite des Projektleiters und anderer Beteiligter mit nicht-formalem Hintergrund kommen.

Die Phasenbeschreibungen, die bei METHONTOLOGY gegeben werden, und die Erläuterungen der in diesem Vorgehensmodell vorgeschlagenen Methoden sind nicht sehr komplex aufgebaut. Auf der anderen Seite kann die Struktur des Modells selbst dadurch unüberschaubar wirken, dass der Ansatz sehr umfassend konzipiert ist und eine große Anzahl von Phasen aufweist, die eine einfache Nachvollziehbarkeit beeinträchtigen.

Die Menge der Vorgehensschritte, die im On-To-Knowledge-Ansatz dargelegt werden, ist dagegen noch übersichtlich und verständlich. Dennoch erfüllt dieser Ansatz das Kriterium der

Einfachheit nur teilweise, weil er viele verschiedene Einflüsse aufnimmt (aus den Bereichen Software Engineering, Wissensmanagement und Ontologieentwicklung) und daher für die Realisierung der Aktivitäten sehr unterschiedliche Methoden vorgeschlagen werden.[717]

Das Vorgehen des Kollaborativen Ansatzes ist sowohl in Hinsicht auf seine äußere Struktur als auch bezüglich der Erläuterungen zu den einzelnen Phasen vergleichsweise simpel gestaltet. Da eventuelle „technische" Details, die etwa die Formulierung und Darstellung der Basis-Ontologie oder auch später die formalsprachliche Implementierung betreffen, außer Acht gelassen werden, ist die Beschreibung der Phasen und ihrer Aktivitäten wenig komplex und durchaus leicht zu erfassen.

5.3.3.3.5 Klarheit

Die dargestellten Ansätze beinhalten kaum formale oder semi-formale Beschreibungen des jeweiligen Vorgehens, dennoch sind die Modelle meist verständlich und eindeutig formuliert und oft durch eine grafische Darstellung des Phasenablaufs verdeutlicht.

Der IDEF5-Ansatz besitzt eine klare Zielsetzung und eine übersichtliche Struktur. Die Aktivitäten in den einzelnen Phasen werden verständlich beschrieben und durch zahlreiche Formblätter mit Beispielen weiter verdeutlicht. Es existiert nur eine natürlichsprachliche Beschreibung der Phasenaktivitäten.

Das Vorgehensmodell des Enterprise-Model-Ansatzes besitzt eine klare Zielsetzung und eine übersichtliche Struktur und die einzelnen Aktivitäten sind allgemein verständlich beschrieben. Leider verweisen die Autoren für detaillierte Ausführungen oft auf andere Quellen.

Die Anforderung der Klarheit wird bei TOVE von Grüninger und Fox vor allem durch das sehr systematisch aufgebaute Phasenmodell umgesetzt, auch die Phasen selbst sind eindeutig formuliert. Besonders bei der Darstellung der Aktivitäten zur Formalisierung der Ontologien verwenden die Autoren zusätzlich formalsprachliche Notationen, um die Eindeutigkeit ihres Ansatzes zu erhöhen.

Auch der METHONTOLOGY-Ansatz präsentiert ein strukturiertes Konzept zur Ontologieentwicklung. Das Grundgerüst dieses Ansatzes ist an den IEEE-Standard für die Entwicklung von Software-Life-Cycle-Prozessen angelehnt[718] und wird durch eine detaillierte und klar formulierte Beschreibung der Methoden erweitert.

Der Grundsatz der Klarheit wird im On-To-Knowledge-Ansatz ebenfalls beachtet: Er beinhaltet ein strukturiertes Phasenmodell mit festgelegtem Umfeld und festgelegter Zielvorgabe. Allerdings ist die Zuordnung der Aktivitäten zu den Phasen (in verschiedenen Veröffentlichun-

717) So werden bspw. für die Erhebung der Anforderungen die Erstellung eines Anforderungsspezifikationsdokuments (aus dem Bereich des Software Engineering) sowie zusätzlich die Formulierung von Kompetenzfragen (ein eher ontologieentwicklungsspezifisches Konzept) empfohlen und beide Dokumente auch später als Referenzrahmen bei der Evaluation herangezogen. Vgl. hierzu Sure, Staab et al. (2004), S. 120 f., und Staab (2002), S. 203 f.

718) Vgl. Fernández López (1999), S. 4.9.

gen) mehrdeutig oder sogar inkonsistent. So wird z. B. die Erstellung von Basis-Ontologien einmal der Phase „Kickoff“ und einmal der „Verfeinerung“ zugeordnet.[719]

Die Vorgehensweise im Kollaborativen Ansatz ist ausschließlich informal, aber eindeutig und verständlich dargestellt und wird durch ein strukturiertes Phasenschema unterstützt. Dennoch können Unklarheiten entstehen, weil die Autoren auf einige Aktivitäten, die Bestandteil des Ansatzes sind, wie z. B. das Vorgehen bei der Entwicklung der Basis-Ontologien, gar nicht oder nur oberflächlich eingehen.

5.3.3.3.6 Werkzeugunterstützung

Da das Gebiet der Ontologieentwicklung für Informatik- und wirtschaftswissenschaftliche Zwecke ein vergleichsweise junges Forschungsfeld ist, erscheint die Auswahl an Werkzeugen für die computerunterstützte Entwicklung von Ontologien begrenzt. Dennoch versuchen einige der Autoren der existierenden Ansätze, Hinweise und Empfehlungen bezüglich einer Software-Umgebung für die Umsetzung ihres Vorgehensmodells zu geben.

Neben den beiden Sprachen zur Ontologieentwicklung verfügt der IDEF5-Ansatz, der auf eine grafische Darstellung verzichtet, über zahlreiche Formblätter, die als einzelne Werkzeuge angesehen werden. Der Ansatz wurde so ausgelegt, dass ohne Computerunterstützung Ontologien entwickelt werden können, sofern man davon ausgeht, die *Elaboration Language* „per Hand“ zu kodieren. Eine Integration der Werkzeuge wird nicht umfassend angestrebt. So gehen bspw. die Ansätze von METHONTOLOGY und On-To-Knowledge mit ihren „Suiten“ in der Mächtigkeit der Werkzeugunterstützung deutlich über diesen Ansatz hinaus.

Uschold und King verweisen bei ihrem Enterprise-Model-Ansatz für die Kodierung der Ontologien und die Dokumentation der Entwicklung auf die Nutzung von Ontolingua[720], obwohl die Aktivitäten des Ansatzes oft nur auf abstrakter Ebene beschrieben werden.

Für den TOVE-Ansatz wäre es wegen seines strukturierten Phasenaufbaus und der vorgeschlagenen formalsprachlichen Notationen nahe liegend, den Entwicklungsprozess durch Computerwerkzeuge zu unterstützen. Diese Hilfsmittel werden aber bei der Darstellung des Ansatzes nicht erwähnt.

Die Autoren von METHONTOLOGY unterstreichen den Nutzen des Einsatzes einer Software-Umgebung für die Ontologieentwicklung[721] und empfehlen für den Import, Export und die Erstellung von Ontologien die Arbeitsplattformen ODE und WebODE[722].

---

719) Vgl. bspw. Lau, Sure (2002), S. 125, für die Zuordnung zur “Kickoff”-Phase und Maedche, Staab et al. (2003), S. 326 f., für die Zuordnung zur Phase der „Verfeinerung“.

720) Ontolingua ist eine Software-Umgebung zur Konstruktion von Ontologien durch verteilt arbeitende Gruppen, die neben einer Bibliothek bestehender Ontologien auch einen Editor für die Generierung und Bearbeitung von Ontologien bereitstellt und diese in verschiedene Sprachen übersetzen kann; vgl. dazu Gruber (1993), S. 203 ff., und Farquhar, Fikes et al. (1996), S. 44.3 ff., sowie Kapitel 4.2.2.1.2, S. 105 f.

721) Vgl. Fernández, Gómez-Pérez et al. (1997), S. 39.

722) Vgl. zu WebODE Arpírez, Corcho et al. (2001).

Im Rahmen des On-To-Knowledge-Ansatzes wird eine umfassende Architektur für die Realisierung einer Entwicklungsumgebung vorgestellt[723], deren Kernbestandteil „OntoEdit“ ist, eine Software zur Erstellung, Bearbeitung, Formalisierung und Visualisierung von Ontologien.

Der Kollaborative Ansatz von Holsapple und Joshi hingegen fokussiert die zwischenmenschliche Diskussion zur gemeinsamen Ontologieentwicklung. Auf mögliche Unterstützung durch spezielle Software-Werkzeuge gehen die Autoren nicht ein. Nur für die Sammlung und Speicherung der Rückmeldungen der verschiedenen Teilnehmer wird der Einsatz einer Datenbank als Werkzeugunterstützung erwähnt.

Insgesamt fällt auf, dass die Vorschläge, die hinsichtlich der Nutzung von Software-Werkzeugen gemacht werden, sich in fast allen Fällen auf die Anwendung in einzelnen Projektphasen beziehen statt auf den gesamten Entwicklungsprozess, wie es zum Beispiel im Software Engineering bereits Praxis ist.

723) Vgl. Sure, Studer (2002), S. 17 ff.

5.3.3.4 Zusammenfassung der Analyseergebnisse

Die folgende Tabelle fasst die Ergebnisse zur Anforderungserfüllung durch die verschiedenen Ansätze zusammen.

Ansatz / *Anforderung*	IDEF5	Enterprise-Model	TOVE	METHONTOLOGY	On-To-Knowledge	Kollaborativer Ansatz
Generizität	+	O	O	+	O	O
Anwendungsbezogenheit	O	– / O	+	O	+	O / +
Begründung Dokumentation	O / +	+	O	+	–	O
Einfachheit	O	O / +	–	O	O	+
Klarheit	O	+	+	+	O	O
Werkzeugunterstützung	O	O	–	O / +	+	–

Tabelle 5: Ergebnisse der Überprüfung der Anforderungen[724]

In einer vergleichbaren Untersuchung aus dem Jahr 2003 kommen der Verfasser und seine Koautorin Apke zu einem ähnlichen Urteil.[725] Hieraus folgern sie die Notwendigkeit, ein spezifisches Vorgehensmodell für das Kompetenzmanagement zu entwickeln, um die Besonderheiten eines Kompetenzmanagementsystems ausreichend zu berücksichtigen, weil Wissen über Wissen erhoben werden soll und nicht auf der Objekt-Ebene verharrt wird.[726]

Fernández López kommt zu dem Urteil, dass im Jahr 1999 der METHONTOLOGY-Ansatz das am weitesten entwickelte Vorgehensmodell zur Entwicklung von Ontologien darstellt.[727]

724) Legende: (-) entspricht *negativ*, (o) entspricht *durchschnittlich*, (+) entspricht *positiv*. Die scheinbare Doppelbewertung soll den Zwischenfall darstellen. Es ergeben sich somit 5 mögliche ordinale Merkmalsausprägungen.

725) Vgl. Apke, Dittmann (2003a), insbesondere S. 59. Dabei wird jedoch bspw. der IDEF5-Ansatz nicht berücksichtigt.

726) Im Ergebnis siehe hierzu auch Apke, Dittmann (2003b) und Apke, Dittmann (2004).

727) Vgl. Fernández Lopéz (1999). Die Untersuchung vergleicht Enterprise-Model-Ansatz, TOVE-Ansatz, METHONTOLOGY-Ansatz, KACTUS und SENSUS. Der Verfasser folgt hier nicht der Auffassung von Fernández López, dass es sich bei SENSUS um ein allgemeines Vorgehensmodell zur Erstellung von Ontologien handelt. Vielmehr handelt es sich um eine Anwendung, aus der auch Ontologien entwickelt werden können. Weil SENSUS jedoch auf eine bestehende Ontologie aufsetzt (genau genommen stellt diese Ontologie die Anwendung im engeren Sinne dar), sieht der Verfasser den Einfluss der Ursprungs-Ontologie als zu hoch an, um noch sagen zu können, dass es sich um die Erstellung „neuer“ Ontologien handelt (siehe hierzu auch Kapitel 5.1.2.3.2.1, S. 149 ff.). Ähnliches gilt für die Betrachtung von KACTUS (siehe hierzu auch Kapitel 5.1.2.3.2.2, S. 150 f.).

5.4 Beispielhafte Konkretisierungen

5.4.1 Allgemeine Grundlagen

An dieser Stelle werden einige spezifische Vorgehensmodelle als beispielhafte Konkretisierungen für die generischen Vorgehensmodelle der Bereiche Software, Knowledge und Ontology Engineering (aus Kapitel 2.1.2.5, S. 31 ff.) vorgestellt. Vornehmlich wird angestrebt, dem Leser die bisher vorgestellten generischen Vorgehensmodelle anhand exemplarischer Konkretisierungen zu verdeutlichen. So soll ermöglicht werden, das in Kapitel 6, S. 193 ff., vorgestellte OntoFMEA-Vorgehensmodell zur Entwicklung von FMEA-Ontologien für die Verwendung in einem Wissensbasierten System wie OntoFMEA (Kapitel 7, S. 252 ff.) hinsichtlich seines Inhalts einzuordnen und zu beurteilen. Neben der Erhöhung der Nachvollziehbarkeit hinsichtlich der Problemstellung dieser Arbeit wird zusätzlich angestrebt, Besonderheiten von spezifischen Vorgehensmodellen gegenüber generischen Vorgehensmodellen weiter herauszuarbeiten. Hieraus resultierende Erkenntnisse sollten bei der Erstellung des OntoFMEA-Vorgehensmodells gegebenenfalls berücksichtigt werden können. Um nicht zu weit vom Fokus dieser Arbeit abzuweichen, werden lediglich einzelne, ausgewählte Vorgehensmodelle aus Kapitel 5.1.2, S. 124 ff., stellvertretend konkreter betrachtet.[728] Für jeden der genannten Bereiche wird zumindest ein konkretisiertes Vorgehensmodell vorgestellt.

Für das Software Engineering wird das V-Modell exemplarisch erläutert. Für das Knowledge Engineering werden die Vorgehensmodelle CommonKads und MIKE und für das Ontology Engineering wird das KOWIEN-Vorgehensmodell vorgestellt. Die Auswahl dieser exemplarischen Konkretisierungen der generischen Vorgehensmodelle wurde für die einzelnen Gebiete aufgrund der folgenden Überlegungen vorgenommen:

- Software Engineering
  Für das Software Engineering wird eine Konkretisierung vorgestellt, die die allgemeine Darstellung des Wasserfall-Ansatzes berücksichtigt: es werden kurz das *V-Modell* und dessen Systematik erläutert. Dieses Vorgehensmodell hat eine große Bedeutung für die Öffentliche Verwaltung (Bund) in der Bundesrepublik Deutschland.[729] Das V-Modell als Vertreter eines Wasserfall-Ansatzes wird exemplarisch vorgestellt, weil die zu erwartenden Entwicklungsmühen eines Wissensbasierten Systems auf der Basis von Ontologien und FMEA-Wissen nicht mit einer so hohen Komplexität verbunden sind, wie sie für den erfolgreichen Einsatz des weiterentwickelten Spiral-Ansatzes notwendig erscheinen.[730] Eine exemplarische Konkretisierung für den Prototyp-Ansatz als verbleibende Alternative[731] scheidet aus, weil für die Entwicklung des Wissensba-

728) Zur Auswahl und zu den der Auswahl zu Grunde liegenden Kriterien der generischen Vorgehensmodelle des Software Engineering, des Knowledge Engineering und des Ontology Engineering siehe Kapitel 5.1.1, S. 120 ff.

729) Siehe hierzu die Ausführungen im folgenden Kapitel.

730) Siehe hierzu die Ausführungen in Kapitel 5.3.1, S. 165 ff.

731) Der sequentielle Software-Life-Cycle-Ansatz wird nicht weiter betrachtet, weil der Wasserfall-Ansatz den „elaborierteren“ von beiden Ansätzen darstellt.

sierten Systems Vorgehensmodelle präferiert werden, die ein explizites Modell einer Domäne berücksichtigen.[732]

- Knowledge Engineering
  Wie bereits erwähnt wurde, werden bei den generischen Vorgehensmodellen des Knowledge Engineering die Modellbasierten Ansätze den Prototyp-Ansätzen vorgezogen.[733] Um Modellbasierte Ansätze eingehender zu verdeutlichen, wird im Folgenden auf zwei bekannte spezifische Ansätze näher eingegangen. Der erste Ansatz (CommonKADS[734]) stellt einen Quasi-Standard[735] zur Entwicklung Wissensbasierter Systeme dar und dient an dieser Stelle zur kurzen Verdeutlichung der Berücksichtigung von expliziten Modellen in der Entwicklung von Wissensbasierten Systemen. Der zweite Ansatz (MIKE) ist aufgrund des Entwicklungshintergrunds von Interesse: MIKE wurde vom Institut für Angewandte Informatik und Formale Beschreibungssprachen in Karlsruhe (AIFB) vorgestellt. Das Institut nimmt im Forschungsfeld „Ontologien" eine starke Position in der Welt ein und gilt in Deutschland als führend.[736] Der Ansatz MIKE kann als Vorläufer für spezifischere Vorgehensmodelle des Ontology Engineering angesehen werden. Die Vorstellung von MIKE bereitet somit den Übergang zu den Vorgehensmodellen des Ontology Engineering im anschließenden Kapitel vor.

- Ontology Engineering
  Das KOWIEN-Vorgehensmodell wird vorgestellt, weil es im Gegensatz zu den untersuchten Ansätzen des Ontology Engineering bereits eine Spezialisierung eines Ontology-Engineering-Vorgehensmodells mit dem Gesamtfokus auf Ontologien und auf eine bestimmte Domäne (Kompetenzmanagement) darstellt. Dies entspricht der Problemstellung der Arbeit, in der ein Wissensbasiertes System auf der Basis von Ontologien für die Domäne der FMEA innerhalb des Qualitätsmanagements eines Unternehmens entwickelt wird.[737]

---

732) Vgl. hierzu Kapitel 5.3.3.1, S. 169. Die spezielle Konkretisierung „V-Modell" ist in etwa vergleichbar mit der Darstellung des OntoFMEA-Vorgehensmodells, auf das hier vorgreifend verwiesen wird (vgl. zum OntoFMEA-Vorgehensmodell insbesondere Kapitel 6, S. 193 ff.).

733) Siehe hierzu den Kapitelverweis in der voranstehenden Fn. Zusätzlich siehe auch Kapitel 5.3.2, S. 168 ff.

734) Zu diesem Vorgehensmodell siehe Schreiber; Wielinga et al. (1994) und Schreiber, Akkermans et al. (2001).

735) Sofern sich in einer jungen Disziplin wie dem Knowledge Engineering und den mit der Jugendhaftigkeit verbundenen Entwicklungssprüngen überhaupt von einem Standard ausgehen lässt, so wird dieser eingeschränkt als „quasi" bezeichnet.

736) Vgl. bspw. Zelewski (2005), S. 177 f.

737) Dem Verfasser ist kein weiteres derartiges „weniger generisches" Vorgehensmodell im Bereich des Ontology Engineering bekannt.

5.4.2 Software Engineering: V-Modell

Aufgrund unkontrollierten Wachstums in der Verwendung von IT und sich verkürzender Innovationszyklen kam es zu zunehmender Unübersichtlichkeit in den IT-Systemen der Bundesverwaltung, so dass sämtliche IT-Ressourcen in einigen Bereichen der Bundesverwaltung vollständig für die Softwarepflege eingesetzt werden mussten und für neue Entwicklungen keine Kapazitäten mehr verfügbar waren.[738] Eine Standardisierungsbemühung – anfänglich angestrebt vom Bundesministerium für Verteidigung – führte zur Formulierung des V-Modells. Das V-Modell legt die Vorgehensweise bei der Softwareentwicklung in seiner vom Bundesministerium des Inneren vorgesehenen Form verbindlich als Standard für den gesamten öffentlichen Bereich fest.

Innerhalb des V-Modells werden vier Submodelle, die miteinander interagieren, unterschieden (siehe Abbildung 38). Neben der *Systemerstellung* wird in Submodelle für *Qualitätssicherung*, *Projektmanagement* und *Konfigurationsmanagement* differenziert. Während Qualitätssicherung, Projektmanagement und Konfigurationsmanagement die begleitenden Aktivitäten in einem Entwicklungsprojekt beschreiben, wird die Entwicklung selbst innerhalb der Systemerstellung vollzogen.[739]

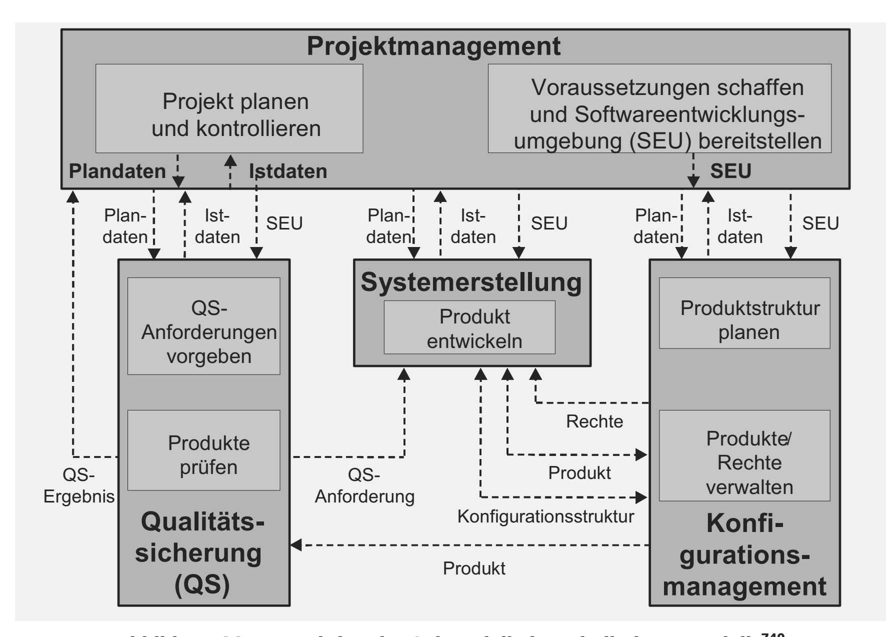


Abbildung 38: Interaktion der Submodelle innerhalb des V-Modells[740]

738) Vgl. Bröhl, Dröschel (1995), S. 15.

739) Vgl. V-Modell (1997), S. 4-1.

740) Das V-Modell zum Download findet sich unter: http://www.informatik.uni-bremen.de/gdpa/, Zugriff am 16.3.2007. Die Abbildung wurde V-Modell (1997), S. 2.9, entnommen.

Das Gesamtmodell lässt sich kurz anhand der folgenden Merkmale charakterisieren:

1. Das Modell erhebt den Anspruch auf Allgemeingültigkeit.
2. Es werden 25 Rollen für Managementaufgaben definiert.
3. Es muss eine Anpassung an konkrete Projektanforderungen erfolgen.
4. Es ist gedacht als Entwicklungsmodell für ein Gesamtsystem, das aus unterschiedlichen Software-Applikationen bestehen kann.

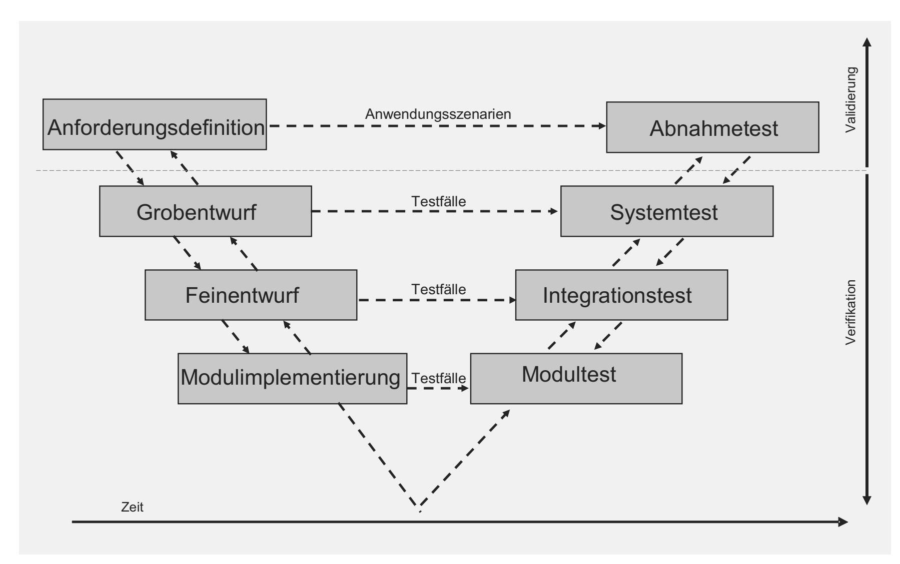


Abbildung 39: Übersicht über das V-Modell[741]

Insgesamt erfolgt die Softwareentwicklung im Zeitverlauf von der Anforderungsdefinition zum Abnahmetest über die beiden übergreifenden Phasen *Verifikation* und anschließende *Validation* (siehe Abbildung 39). Hierbei ergibt sich in der Darstellung das charakterisierende „V". Es entspricht in der Grundstruktur dem zuvor in Kapitel 5.1.2.1.2 beschriebenen Wasserfall-Modell, geht jedoch durch seine zahlreichen semi-formalen Vorgaben (insbesondere zu den Tests)[742] darüber hinaus. Die Phasen *System-Anforderungen* und *Software-Anforderungen* werden zur Phase *Anforderungsdefinition* zusammengefasst. *Analyse* und *Entwurf* werden innerhalb der Phasen *Grob-* und *Feinentwurf* abgehandelt. Die *Kodierung* entspricht der *Modulimplementierung*. Die Phase *Testen* wird innerhalb des V-Modells besonders berücksichtigt, so finden sich *Modultest, Integrationstest, Systemtest* und *Abnahmetest* im V-Modell.

741) Vgl. http://www.informatik.uni-bremen.de/uniform/gdpa/def/def_v/V_MODEL.htm, Zugriff am 16.03.2007.

742) Vgl. bspw. V-Modell (1997), S. 4.3 ff., für das Submodell *Systemerstellung.*

Es wird deutlich, dass das V-Modell einen Verwaltungsaufwand mit sich bringt, der für einzelne Softwareentwicklungen oft ungeeignet scheint, weil aufgrund der Komplexität des V-Modells übermäßig viele Ressourcen durch die Anwendung des V-Modells gebunden werden. Es erscheint daher zweckmäßiger für die Entwicklung des OntoFMEA-Vorge-hensmodells auf „kompaktere" Vorgehensmodelle zurückzugreifen.

5.4.3 Knowledge Engineering: CommonKADS und MIKE

5.4.3.1 CommonKADS

KADS (Knowledge Analysis and Documentation System) und die Weiterentwicklung zu CommonKADS stellen einen bekannten Ansatz für ein Modellbasiertes Vorgehensmodell des Knowledge Engineering dar.[743] CommonKADS besinnt sich auf Elemente aus der objektorientierten Programmierung und verwendet Elemente der UML-Notation (bspw. Klassendiagramme, Aktivitätsdiagramme und Zustandsdiagramme). Der Entwicklungsfokus wird auf die frühen Phasen im Software-Life-Cycle gelegt, d. h., die Wissensakquisition wird deutlich stärker gewichtet als die Implementierung eines Wissensbasierten Systems.

Die Abbildung 40 zeigt die Modellsammlung, d. h. die einzelnen Modelle hinsichtlich ihrer obersten Ebene, die zusammen den CommonKADS-Ansatz bilden. Nach Erstellung eines *Organisations-Modells*, eines *Aufgaben-Modells* und eines *Agenten-Modells* auf der Kontext-Ebene werden innerhalb dieser Modelle die besonders wissensintensiven Abläufe eruiert und anschließend in einem *Wissens-Modell* und einem *Kommunikations-Modell* auf der Konzept-Ebene analysiert. Für das *Entwurfs-Modell* auf der Artefakte-Ebene werden Anforderungen für Schlussfolgerungsfunktionen aus dem Wissens-Modell und Anforderungen zur Spezifikation für Interaktionsfunktionen aus dem Kommunikations-Modell abgeleitet.[744] Innerhalb des Organisations-Modells erfolgt die Spezifizierung der Anforderungen, die ein Wissensbasiertes System in einer Organisation erfüllen soll. Innerhalb des Agenten-Modells werden die Funktionen des Wissensbasierten Systems, abgeleitet aus dem Organisations-Modell, sowie die beteiligten (sowohl menschlichen als auch artifiziellen) Agenten definiert. Anschließend wird das Agenten-Modell in eine Anzahl von Aufgaben zerlegt und als Aufgaben-Modell dargestellt.[745] Das benötigte Problemlösungswissen, das zur Erfüllung der Aufgaben gebraucht wird, wird im Wissens-Modell festgelegt. Die Zerlegung der Aufgaben und Verteilung auf die Agenten wird im Kommunikations-Modell vorgenommen. Die Beschreibung der

743) Zu diesem Vorgehensmodell vgl. Schreiber, Akkermans et al. (2001).

744) Der Bereich des Ontology Engineering findet in diesem Zusammenhang innerhalb des *Wissens-Modells* (Knowledge Model) seine Berücksichtigung. Schreiber, Akkermans et al. verwenden den Begriff „Ontologien" jedoch abweichend zu dem hier gebrauchten Verständnis. Vielmehr sprechen sie von „domain schemas" (vgl. hierzu insbesondere Schreiber, Akkermans et al. (2001), S. 91). Ontologien stellen für die Autoren in diesem Zusammenhang „generalized domain schemas" dar (vgl. Schreiber, Akkermans et al. (2001), S. 331). Die Autoren bleiben jedoch eine Antwort schuldig, worin ihr Unterschied zwischen generalisierten und nicht generalisierten Domänenschemata liegt.

745) Die beschriebenen Zusammenhänge zwischen den Modellen der Kontext-Ebene finden sich nicht in der Abbildung wieder. Um die Übersichtlichkeit zu wahren, wurde die Abbildung gemäß ihrer Quelle (siehe hierzu die folgende Fn.) dargestellt, d. h. es werden nur die Beziehungen (Kanten) zwischen der Kontext-, Konzept- und Artefakte-Ebene dargestellt.

Repräsentation, mit der das gewünschte Verhalten eines Wissensbasierten Systems erreicht werden soll, wird im Entwurfs-Modell bearbeitet.

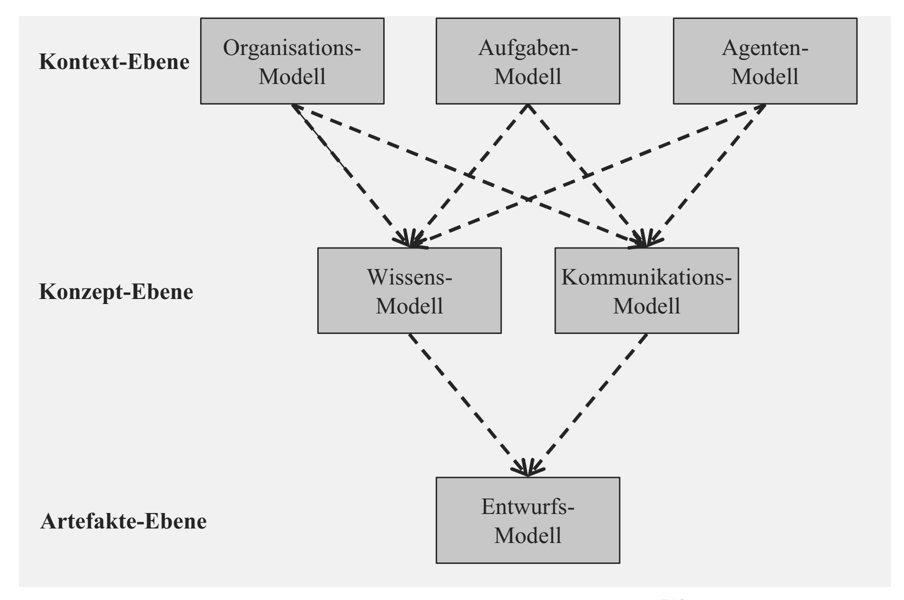


Abbildung 40: Die CommonKADS Model Suite[746]

Es wird deutlich, dass die Phase der Wissensakquisition und damit die spezifischen Möglichkeiten der Akquisition von Wissen innerhalb einer festgelegten Domäne starke Berücksichtigung auch innerhalb des OntoFMEA-Vorgehensmodells finden muss. Dagegen kann die eigentliche Implementierung des Wissensbasierten Systems ein wenig in den Hintergrund treten. Ferner zeigt sich, dass für die Entwicklung eines Wissensbasierten Systems ein Ebenenkonzept vorteilhaft erscheint, um die Komplexität bei der Repräsentation von Wissen bewältigen zu können, denn die Repräsentation stellt selbst bereits Wissen dar.

5.4.3.2 MIKE

Als Vorgehensmodell zur systematischen Entwicklung von Wissensbasierten Systemen wurde MIKE (Modellbasiertes und Inkrementelles Knowledge Engineering) 1993 von Angele, Fensel, Landes, Neubert und Studer vorgestellt.[747]

Vergleichbar mit dem Spiralmodell von Boehm[748] wird der Entwicklungsprozess für ein Wissensbasiertes System zyklisch und inkrementell aufgebaut.[749] Ähnlich wie im vorgestellten

746) Vgl. Schreiber (2001), S. 18 f.

747) Vgl. Angele, Fensel et al. (1993).

748) Siehe Kapitel 5.1.2.1.4, S. 129.

749) Vgl. Pierlein (1995), S. 165.

CommonKADS *Wissens-Modell* wird ein konzeptuelles Modell auf der implementierungsunabhängigen Ebene anhand von drei Wissensarten entwickelt:[750]

1. Die domänenspezifischen Konzepte, Attribute und Relationen werden als Wissen in der *Domänenebene* modelliert.

2. Die Problemlösungsmethode wird in der *Inferenzebene* abgelegt. Die notwendigen Inferenzen sowie ihre gegenseitigen Einflüsse werden in dieser Ebene spezifiziert.

3. Die *Aufgabenebene* enthält das Wissen, um eine spezifische Aufgabe lösen zu können. Insbesondere die Reihenfolgen der Inferenzen, die in einem Problemlösungsprozess anzuwenden sind, werden hier festgelegt.

Der Prozess des Knowledge Engineering gliedert sich in MIKE in die Phasen *Wissensakquisition, Design, Implementierung* und *Evaluierung*.[751]

1. Wissensakquisition
   Die Phase der Wissensakquisition wird in MIKE in drei Unterphasen gegliedert.
   Erstens umfasst die *Aufgabenanalyse* die Festlegung des zu entwickelnden Gesamtsystems und die darin enthaltene Einbettung des Wissensbasierten Systems. Es werden die funktionalen Anforderungen an das Wissensbasierte System erhoben.
   Zweitens beinhaltet die *Modellkonstruktion* (auch Modell*bildung* genannt in Angele, Fensel et al. (1995), S. 19) die zu erstellenden Modelle innerhalb des Vorgehensmodells. Sie wird wiederum differenziert in *Wissenserhebung, Interpretation, Formalisierung* und *Operationalisierung*. Die Wissenserhebung in MIKE umfasst die Grobererhebung und die Feinerhebung. Das Erhebungs-Modell stellt das Ergebnis der Wissenserhebungsphase dar. Es besteht vornehmlich aus Protokollen, die das Wissen aus Gesprächen, Beobachtungen etc. mit Experten beinhalten. Während der Phase der Interpretation werden das Dokumentations-Modell und das Struktur-Modell des zu repräsentierenden Wissens entwickelt. Erhebungs-Modell, Dokumentations-Modell und Struktur-Modell sind miteinander verbunden.[752] Das Wissens-Modell wird in der Formalisierungsphase in der formalen Wissensrepräsentationssprache KARL[753] beschrieben. In der Phase der Operationalisierung wird das formalisierte Modell in das Wissensbasierte System implementiert. Im Vordergrund dieser ersten Implementierung steht dabei die Überprüfung der Verarbeitbarkeit des formalisierten Modells.
   Drittens wird das Modell der Expertise in der *Modellevaluationsphase* in einem Prototyp-Ansatz ein erstes Mal evaluiert.[754]

---

750) Vgl. Angele, Fensel et al. (1993), S. 6 ff., und Angele, Fensel et al. (1995), S. 17 ff.

751) Vgl. Angele, Fensel et al. (1995), S. 19, und Pierlein (1995), S. 168.

752) Vgl. Pierlein (1995), S. 173.

753) KARL basiert auf F-Logic und damit auch auf Prädikatenlogik (vgl. Angele, Fensel et al. (1995), S. 19 bzw. S. 17). Weitere Informationen zu KARL finden sich in Fensel (1995) und in Fensel, angele et al. (1998).

754) Vgl. Angele, Fensel et al. (1995), S. 20.

2. Design
   Die Design-Phase umfasst zusätzlich die Berücksichtigung nicht-funktionaler Anforderungen (bspw. Effizienz der Ausführung der Software und Wartbarkeit), dabei soll die Struktur des Wissens-Modells weitgehend erhalten bleiben, um die Anschlussfähigkeit des Wissens zu gewährleisten. Wiederum werden drei Schritte durchlaufen: Auf die Aufgabenanalyse folgt die Modellkonstruktion (als Ergebnis liegt das so genannte Design-Modell vor) und anschließend die Modellevaluation mittels Entwicklung eines Prototyps.

3. Implementierung
   An die Entwurfsphase schließt sich die Phase der endgültigen Implementierung des Wissensbasierten Systems mit Berücksichtigung sowohl der funktionalen als auch der nicht-funktionalen Anforderungen an. Hierbei legen sich die Autoren auf eine Implementierung des Systems in C++ auf Basis des Design-Modells fest.[755]

4. Evaluierung
   Zum Schluss erfolgt die Abschlussevaluation, die das lauffähige System abschließend einer (Gesamt-)Bewertung unterzieht.

Die Autoren beschreiben ihr inkrementelles Vorgehen als einen sanften, in mehreren Schritten erfolgenden Übergang von informalen Wissensprotokollen zu einem endgültigen System.[756] Deutlich wird die explizite Berücksichtung von Modellen zur Abbildung von Wissen.

5.4.4 Ontology Engineering: KOWIEN-Ansatz

Im Rahmen des Forschungsverbundprojekts KOWIEN[757] (Kooperatives Wissensmanagement in Engineering Netzwerken) wurde ein Vorgehensmodell vorgestellt, mit dessen Hilfe eine strukturierte und systematische Entwicklung von Kompetenz-Ontologien für betriebliche Kompetenzmanagementsysteme ermöglicht wird. Ziel des Modells ist die Beschreibung eines systematischen Prozesses zur Konstruktion von Kompetenz-Ontologien (inkl. deren Implementierung) mit Hilfe eines Vorgehensmodells.

Das KOWIEN-Vorgehensmodell lässt sich in fünf Hauptphasen und zwei phasenübergreifende Unterstützungsleistungen gliedern.[758] Die fünf Hauptphasen setzen sich aus der *Anforderungsspezifizierung*, der *Wissensakquisition*, der *Konzeptualisierung*, der *Implementierung* und der *Evaluation* zusammen. Als phasenübergreifende Unterstützungsleistungen werden *Projektmanagement* und *Dokumentation* aufgefasst.

Die Ontologieentwicklung selbst beginnt mit der *Spezifizierung der Anforderungen*, die von der Ontologie erfüllt werden müssen. Hierzu zählen neben der Definition von Kriterien, die

755) Vgl. Angele, Fensel et al. (1995), S. 20.

756) Vgl. Angele, Fensel et al. (1995), S. 20.

757) Vgl. Zelewski, Alan et al. (2005).

758) Siehe hierzu auch Abbildung 43, S. 199.

als Richtlinien während des Designs sowie als Referenzrahmen bei der Evaluation der erstellten Ontologie dienen, auch die Festlegung der zukünftigen Anwendungsbereiche und die Identifizierung der Endbenutzer.

Bei der *Wissensakquisition* werden anschließend Artefakte, die über die Verteilung des Wissens im Unternehmen und über vorhandene Kompetenzen Auskunft geben, erfasst.

Dieses Wissen wird im Rahmen der *Konzeptualisierung* strukturiert und verarbeitet. Dabei werden zunächst mit Hilfe von Domänenexperten die für ein Kompetenzmanagementsystem wichtigen Begriffe identifiziert, in Form einer Taxonomie hierarchisch gegliedert und durch Attribute und Relationen beschrieben.[759]

Nachdem diese Konzeptualisierung nur informal und modellhaft ist, wird bei der *Implementierung* eine Sprache ausgewählt und die Konzeptualisierung in eine formale Repräsentation transformiert. Die Semantik dieser Spezifikation wird dann einerseits durch Integritätsregeln zur Einschränkung der Interpretations- und Verknüpfungsmöglichkeiten der Begriffe und andererseits durch Inferenzregeln, die Schlussfolgerungen aus vorhandenem Wisssen ermöglichen, festgelegt.

Vor ihrem Einsatz in den Anwendungsbereichen soll eine gründliche *Evaluation* der resultierenden Ontologie erfolgen; dabei wird zusammen mit den Benutzern die Erfüllung der zuvor aufgestellten Benutzeranforderungen überprüft und ihre Anwendbarkeit im späteren Systemumfeld getestet.

Während des gesamten Entwicklungsprozesses werden die dabei erzielten Ergebnisse sowie die getroffenen Entscheidungen und ihre Grundlagen *dokumentiert*, um sowohl die Konstruktion einer Ontologie selbst als auch eine spätere Wissenswiederverwendung zu unterstützen. Weiterhin ist zu beachten, dass zusätzlich zum eigentlichen Entwicklungsprozess Projektplanungs- und -steuerungsaktivitäten durchzuführen sind, um bspw. den Budgetumfang zu bestimmen und zu kontrollieren. Diese grundsätzlichen Aufgaben des *Projektmanagements* werden in dem hier dargestellten Vorgehensmodell berücksichtigt. Sie sind jedoch nicht ontologieentwicklungsspezifisch, sondern stellen sich allgemein als Anforderung bei Softwareentwicklungsprojekten.[760]

Es wird bei einem domänenspezifischen Vorgehensmodell deutlich, dass für den praktischen Einsatz ein Vorgehensmodell an die Erfahrungswerte und Erwartungen der Benutzer ange-

759) Entsprechend den üblichen Definitionen von Ontologien (siehe Kapitel 2.1.2.4, S. 28 ff.) ist eine Dominanz der taxonomischen Strukturierung keineswegs zwingend. Es zeigt sich jedoch, dass es aus Gründen der leichteren Nachvollziehbarkeit durch Benutzer aus der betrieblichen Praxis sinnvoll ist, mit einer taxonomischen Strukturierung zu beginnen, wenn das Vorgehensmodell in der Praxis auf Akzeptanz stoßen soll.

760) Hierfür wurden bereits Standards, wie etwa der IEEE-Standard 1074:1995, verfasst und auch akzeptiert. Der IEEE-Standard 1074:1995 (vgl. IEEE 1074:1995) beschreibt einen „typischen" Softwareentwicklungsprozess mit den dabei durchzuführenden Aktivitäten und möglichen einsetzbaren Methoden (vgl. auch Fernández López (1999), S. 4.2). Der Entwicklungsprozess wird gemäß IEEE-Standard unterteilt in Unterprozesse des Softwarelebenszyklusmodells, des Projektmanagements sowie in softwareentwicklungsorientierte und integrale Prozesse.

passst sein muss. Für den praktischen Einsatz muss eine Balance gefunden werden zwischen einem wissenschaftlichen (konzeptionellen) und einem praktischen (durchführbaren) Anspruch an ein Vorgehensmodell. Oftmals erwarten die Benutzer eine detaillierte Hilfestellung durch die Methoden und Werkzeuge des Vorgehensmodells ohne dabei hinterfragen zu wollen, aus welchen Gründen eine bestimmte Aktivität durchzuführen ist.

Die besondere Bedeutung des KOWIEN-Vorgehensmodells für das hier vorgestellte Onto-FMEA-Vorgehensmodell zur Entwicklung von FMEA-Ontologien wird im folgenden Kapitel weiter herausgearbeitet.

6 OntoFMEA-Vorgehensmodell

In diesem Kapitel wird das OntoFMEA-Vorgehensmodell vorgestellt, mit dessen Hilfe die FMEA-Ontologie entwickelt wird, die schließlich im Prototyp „OntoFMEA“ das Herzstück des Wissensbasierten Systems bildet. Zunächst werden einige Überlegungen zur FMEA-Ontologieentwicklung, die sich im Vorgehensmodell widerspiegeln, beschrieben. Anschließend wird das Vorgehensmodell in seiner Gesamtdarstellung und den einzelnen Phasen und phasenübergreifenden Unterstützungsleistungen vorgestellt.

6.1 Ausgangslage für die Vorgehensmodellentwicklung

Zu Beginn aller Überlegungen stellt sich die Suche nach der Begründung zur „Zusammenführung“ von FMEA und Ontologie, die in ihrer Argumentation außerhalb eines durch die Aufgabenstellung von außen vorgegebenen Seins ansetzt. Als Begründung für die Verwendung der FMEA zur Generierung von Ontologien sprechen die Faktoren:

- eine FMEA beinhaltet eine systematische Vorgehensweise,
- die FMEA grenzt einen Untersuchungsgegenstand ab und erzeugt eine Domäne,
- zahlreiche Wissensträger werden bereits bei der FMEA-Durchführung involviert und
- die Arbeit in interdisziplinären Teams bildet umfangreiches Wissen ab.

Gerade das gemeinschaftliche Erarbeiten einer FMEA im Team zwingt die Mitglieder, sich auf einen gemeinsamen Wissensstand und die Verwendung einer gemeinsamen Sprache zu einigen. Die Verwendung einer gemeinsamen Sprache wird als hinreichende Bedingung für eine Ontologie angesehen.[761] Insbesondere wird dadurch sichergestellt, dass Konzepte in gleicher Weise Verwendung finden, d. h. es herrscht Konsens bezüglich der Konzepte einer Domäne. Durch die Verwendung von FMEA-Formularen zur Ontologieentwicklung kann sichergestellt werden, dass, obwohl unter Umständen nur sehr wenige Personen an der Entwicklung direkt beteiligt sind, die gemeinsame Begriffswelt vieler Personen maschinenlesbar zur Verfügung gestellt wird.

761) Siehe die Definitionen zur Ontologie auf S. 28 ff.

6.2 Grundlagen des OntoFMEA-Vorgehensmodells

6.2.1 Zweck des Vorgehensmodells

Der Zweck des hier beschriebenen Vorgehensmodells lässt sich so umschreiben:

Das Wissen, das in FMEA-Anwendungen akquiriert wurde, soll wiederverwendet werden. Ein Wissensbasiertes System auf der Grundlage von Ontologien soll zur FMEA-gestützten Wissensakquisition, -aufbereitung und -wiederverwendung zum Einsatz gebracht werden. Ein Vorgehensmodell soll die systematische Entwicklung der FMEA-Ontologien sicherstellen.

Der Benutzer des OntoFMEA-Vorgehensmodells soll in die Lage versetzt werden, eigenständig aus vorhandenen FMEA-Formularen und bei Kenntnis des Instruments FMEA eine FMEA-Ontologie zu entwickeln, die schließlich in einem Wissensbasierten System Anwendung finden kann.

6.2.2 Modellierungsrahmen

Die Entwicklung einer Ontologie erfolgt immer unter einem spezifischen Modellierungsrahmen.[762] Aus einem solchen Rahmen lassen sich erste Schritte für ein Vorgehen destillieren.[763] Ein Modellierungsrahmen, der für diese Arbeit als gültig erachtet wird, lässt sich wie folgt umschreiben (siehe auch Abbildung 41):

- Bereits der Anwendungsbereich[764] der FMEA und das Instrument FMEA selbst werden als Modell aufgefasst; dies ist bei einer vorliegenden durchgeführten FMEA als Arbeitsgrundlage unmittelbar einsichtig, weil die Bestandteile der FMEA immer als Modelle aufgefasst werden können. Bspw. entspricht das untersuchte System (Maschine) lediglich der Untersuchung eines Modells der tatsächlich existierenden Maschine, bei der zumeist lediglich die Hauptfunktionen dargestellt werden, d. h., es findet eine verkürzte Sicht auf die Maschine statt. Das zu repräsentierende Wissen liegt als internes informales Modell vor, weil das Modell in den Ausführungen einer FMEA lediglich schwach strukturiert vorliegt.

- Das zu repräsentierende Wissen, im Fall dieser Arbeit eine vorliegende durchgeführte FMEA, wird zunächst durch den Entwickler konzeptualisiert. Als Ergebnis liegt ein externes informales Modell vor.

- Der Entwickler formalisiert auf unterschiedlichen Abstraktionsebenen diese Konzeptualisierung. Als Ergebnis liegt ein externes formales Modell vor.

---

762) Für die folgenden Ausführungen können Ontologien vereinfacht auch als (Wissens-)Modelle angesehen werden.

763) Diese Destillation erfolgt im nächsten Kapitel.

764) Der Anwendungsbereich einer durchgeführten FMEA entspricht dem untersuchten System (Untersuchungsgegenstand).

- Die Abstraktionsebenen des modellierten Wissens einer durchgeführten FMEA lassen sich in vier Ebenen unterscheiden (Meta-Meta-Ebene, Meta-Ebene, Klassen-Ebene und Instanzen-Ebene, s. u.).

- Das formalisierte Wissen (als explizites Modell) existiert nach der Implementierung im Computer als formales internes Modell[765].

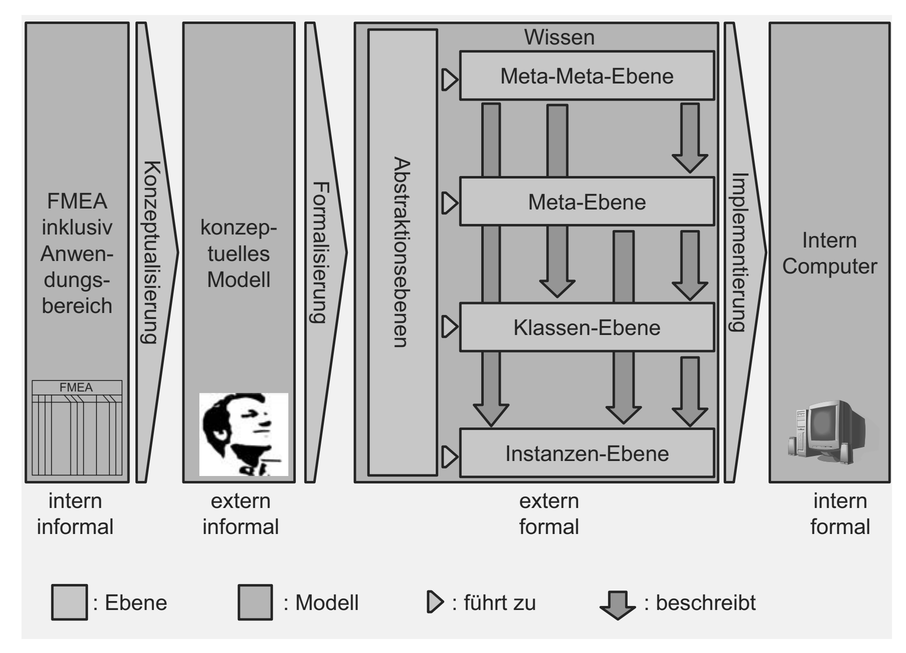


Abbildung 41: Modellierungsrahmen für ein Wissensbasiertes System

Die in der Abbildung 41 dargestellten vier Abstraktionsebenen dienen als Gliederungspunkte für die Darstellung der Explikation des Wissens einer FMEA für die Entwicklung von Ontologien. Die jeweils oberen Ebenen beeinflussen die darunter liegenden Ebenen, indem sie zur Beschreibung der jeweiligen Ebene die Modellierungselemente vorgeben. So beeinflusst die Meta-Meta-Ebene die drei tiefer liegenden Ebenen Meta-Ebene, Klassen-Ebene und Instanzen-Ebene.[766]

765) Das computerinterne Modell lässt sich vereinfacht ausgedrückt als die Aneinanderreihung von Bits vorstellen, die bei ihrer Verarbeitung durch den Computer nicht vom Benutzer eingesehen werden kann. Dabei wird bspw. ein explizites Modell, das mittels einer Inferenzmaschine ausgewertet werden soll, gemäß der Vorgaben der Inferenzmaschine so aufbereitet und gegebenenfalls im Hauptspeicher abgelegt, dass ein Bearbeitungsalgorithmus Schlussfolgerungen hierauf durchführen kann.

766) Auf die Bedeutungen und Funktionsweisen der einzelnen Abstraktionsebenen wird im nächsten Kapitel eingegangen.

6.2.3 Transformation des Modellierungsrahmens zu einem Vorgehen

Der Modellrahmen für die Entwicklung eines Wissensbasierten Systems dient zur Veranschaulichung der Darstellung der Zusammenhänge zwischen FMEA und Ontologien.[767] Aus den Zusammenhängen wird das Vorgehensmodell entwickelt.

Der Ansatz zur Entwicklung einer FMEA-Ontologie basiert in der Struktur auf der Gliederung der Abstraktionsebenen des Wissens aus der Abbildung 41, S. 195. Die Meta-Meta-Ebene, Meta-Ebene, Klassen-Ebene und Instanzen-Ebene bilden das Grundgerüst und den Ausgangspunkt für die Zusammenführung der Explikationen des Wissens der Instrumente FMEA und Ontologien. Abbildung 42 auf der nächsten Seite umreißt diesen Ansatz zur Entwicklung einer FMEA-Ontologie.

Die *Meta-Meta-Ebene* beeinflusst die darunter liegende Meta-Ebene, indem sie die Modellierungsprimitive vorgibt, mit deren Hilfe die Meta-Ebene beschrieben wird. Eine Ontologie auf Basis von FMEA-Wissen soll entwickelt werden, d. h., die Modellierungsprimitive, die durch die Ontologierepräsentationssprache auf der Meta-Meta-Ebene vorgegeben werden, beeinflussen die Modellierungsprimitive der FMEA auf der Meta-Ebene. Auf der Meta-Meta-Ebene werden die Modellierungsprimitive der Repräsentationssprache für Ontologien und damit für die FMEA festgelegt.

Auf der *Meta-Ebene* finden sich die Modellierungsprimitive des Instruments FMEA, insbesondere vorgegeben durch das Formblatt. Es lässt sich feststellen, dass auf der Meta-Ebene die strukturierenden Primitive der FMEA festgelegt werden.

Auf der *Klassen-Ebene* wird das generische Wissen einer FMEA repräsentiert. Es wird auf dieser Ebene das Struktur- und Funktionswissen in Konzepten und in Relationen abgebildet.

Auf der untersten Ebene (*Instanzen-Ebene*) wird das Wissen instanziiert. Hierzu wird das eigentliche Wissen (Objektwissen) einer FMEA, das dem spezifischen Anwendungsbereich einer durchgeführten FMEA entstammt, abgelegt.

767) Einen ähnlichen Aufbau verwenden bspw. Angele, Fensel et al., um die Repräsentationsebenen ihres Vorgehensmodells MIKE (vgl. Kapitel 5.4.3.2, S. 188) zu veranschaulichen und zu systematisieren. Es wird jedoch lediglich in vertikaler Richtung betrachtet: das Wissen eines Experten, das repräsentiert werden soll, wird in einem Erhebungsmodell abgebildet, anschließend wird dieses Wissen interpretiert und in einem Strukturmodell abgebildet, hieran schließt sich mittels Formalisierung die Darstellung eines Modells der Expertise an. Auf Basis dieses Expertisemodells wird ein Designmodell entworfen, das anschließend in der Implementierungsphase umgesetzt wird und damit als Wissensbasis eines Wissensbasierten Systems vorliegt (vgl. Angele, Fensel et al. (1995), S. 17).

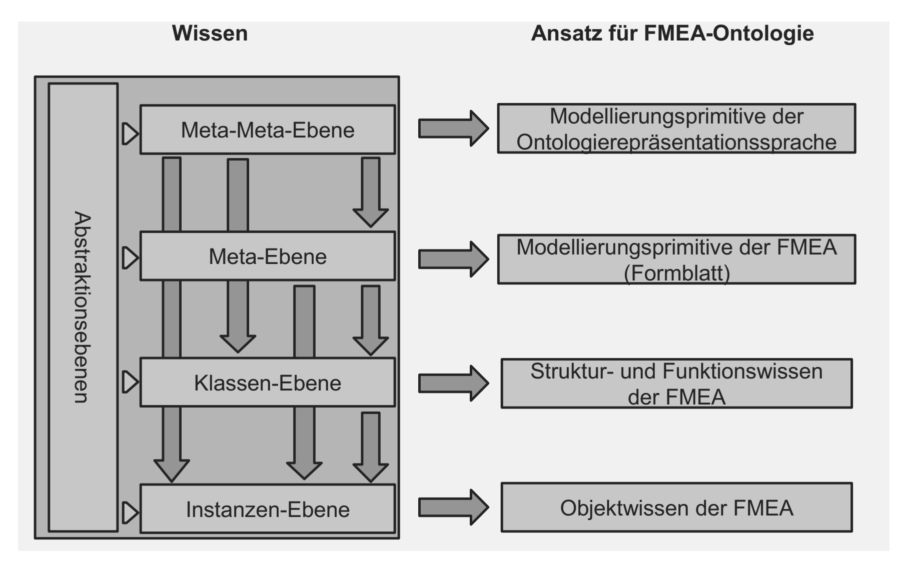


Abbildung 42: Ansatz zur Entwicklung einer FMEA-Ontologie

Durch das Top-Down-Wirken der jeweiligen Ebene ergibt sich eine erste rudimentäre Schrittfolge bei der Entwicklung von Ontologien mittels FMEAs:

1. Eine Sprache zur Wissenrepräsentation wird festgelegt (Modellierungsprimitive der Ontologierepräsentationssprache).

2. Die Hauptkonzepte einer FMEA (Formblatt-Wissen) werden beschrieben (Modellierungsprimitive der FMEA).

3. Die generischen Konzepte und Relationen, die das Wissen einer FMEA spezifisch gliedern, werden beschrieben (Struktur-, Funktionswissen der FMEA).[768]

4. Die Instanzen von Relationen und Konzepten werden festgelegt (Objektwissen der FMEA).

Da im engeren Sinne die Instanzen nicht Teil der Ontologie sind, werden nur die ersten drei Schritte später im Vorgehensmodell wiederzufinden sein.[769]

---

768) Das Wissen einer durchgeführten FMEA, d. h. vornehmlich das Wissen aus den ausgefüllten Zeilen eines FMEA-Formblatts, wird als spezifisches Instanzenwissen abgelegt. Jedoch wird solches Wissen, das eine generische Bedeutung besitzt in der Form, dass es als Ontologie einer Wiederverwendung zugeführt werden soll, aus dem Instanzenwissen abgeleitet und der Klassen-Ebene zugeordnet, um eine Wiederverwendung zu erreichen. Für eine detaillierte Vorstellung dieser Konzeption siehe Kapitel 7.4, S. 244 ff.

769) Siehe hierzu auch Kapitel 2.1.2.4, S. 28 ff.

6.3 Vorgehensmodellübersicht

6.3.1 Einführung

Das im Folgenden vorgestellte OntoFMEA-Vorgehensmodell wird als spezifisches Vorgehensmodell dem Bereich des Ontology Engineering zugeordnet. Es dient dem Zweck, eine Ontologie für die Domäne des betrieblichen Qualitätsmanagements zu entwickeln. Die Entwicklung einer Domänen-Ontologie wird als ein Entwicklungsprojekt von niedrigerer Komplexität angesehen bspw. im Gegensatz zur Entwicklung eines Betriebssystems oder eines vollständigen Wissensbasierten Systems. Nachdem zunächst die oberste Aggregationsstufe des Vorgehensmodells zur Übersicht vorgestellt wird, folgt zum Ende dieses Kapitels eine Diskussion der Einflüsse aus den untersuchten Vorgehensmodellen des Kapitels 5.

6.3.2 OntoFMEA-Vorgehensmodellübersicht Top-Level

Hinsichtlich der Gliederung auf makroskopischer Ebene (Top-Level) orientiert sich das OntoFMEA-Vorgehensmodell am KOWIEN-Vorgehensmodell[770]. Es lässt sich analog zum KOWIEN-Vorgehensmodell in fünf Hauptphasen und zwei phasenübergreifende Unterstützungsleistungen gliedern (vgl. Abbildung 43). Die fünf Hauptphasen sind die *Anforderungsspezifizierung*, die *Wissensakquisition*, die *Konzeptualisierung*, die *Implementierung* und die *Evaluation*. Als phasenübergreifende Unterstützungsleistungen werden ebenfalls *Projektmanagement* und *Dokumentation* berücksichtigt.

Während des gesamten Entwicklungsprozesses werden die erzielten Ergebnisse sowie die getroffenen Entscheidungen und ihre Grundlagen *dokumentiert*, um sowohl die Konstruktion einer FMEA-Ontologie selbst als auch eine spätere Wissenswiederverwendung zu unterstützen. Weiterhin ist zu beachten, dass zusätzlich zum eigentlichen Entwicklungsprozess Projektplanungs- und -steuerungsaktivitäten durchzuführen sind, um bspw. den Budgetumfang zu bestimmen und zu kontrollieren. Diese grundsätzlichen Aufgaben des *Projektmanagements* werden in dem OntoFMEA-Vorgehensmodell berücksichtigt. Sie sind jedoch nicht als ontologieentwicklungsspezifisch anzusehen, sondern stellen sich allgemein als Aufgaben bei Softwareentwicklungsprojekten.

Die Ontologieentwicklung selbst beginnt mit der Spezifizierung der Anforderungen, die von der FMEA-Ontologie erfüllt werden müssen. Die *Anforderungsspezifizierung* endet mit dem Vorliegen einer Anforderungsspezifikation.

Bei der *Wissensakquisition* werden anschließend die Artefakte „FMEA-Formulare" und „Wissensträgerkarte" erfasst. Mit Hilfe der Wissensträger lässt sich das Wissen in den FMEA-Formularen in eine Ontologie transferieren. Die Wissensträger können zu spezifischen Annahmen, die in den Formblättern implizit enthalten sind, Auskunft geben. So wird den Ontologieentwicklern das Verständnis über die Domäne ermöglicht.

770) Siehe hierzu Kapitel 5.4.4, S. 190 ff.

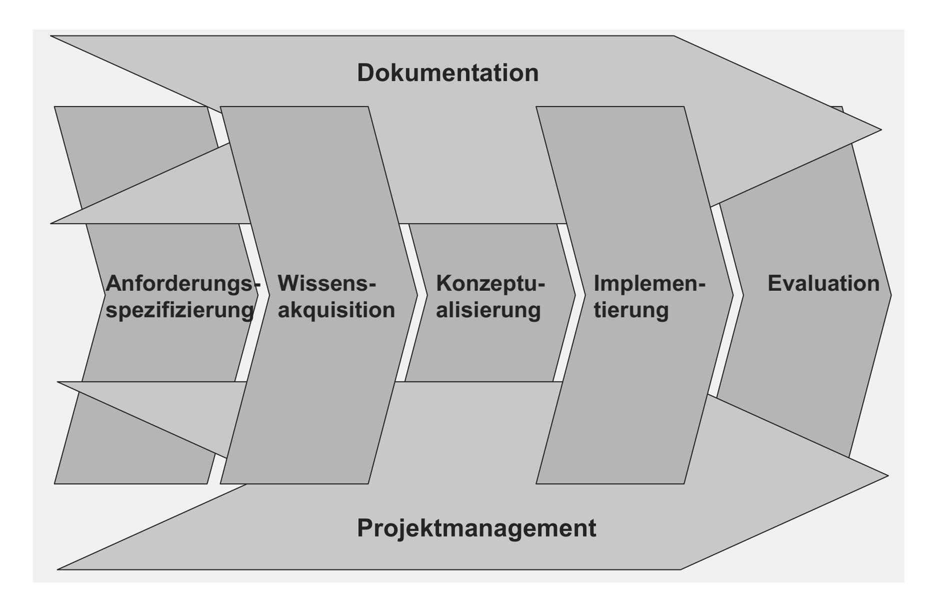


Abbildung 43: Vorgehensmodellübersicht Top-Level

Das zu repräsentierende Wissen wird im Rahmen der *Konzeptualisierung* strukturiert und verarbeitet. Nach der Festlegung einer Repräsentationssprache werden zunächst die Hauptkonzepte identifiziert und mittels Relationen und Regeln näher beschrieben. Anschließend werden Taxonomien zu Systemelementen, Funktionen und Maßnahmen aufgestellt. Zum Schluss werden mögliche weitere Konzepte, Relationen und Regeln analysiert und festgelegt.

Nachdem diese Konzeptualisierung nur informal vorliegt, wird bei der *Implementierung* unter Verwendung der in der Phase der Wissensakquisition ausgewählten Ontologierepräsentationssprache die Konzeptualisierung in eine formale Repräsentation transformiert.

Vor ihrem Einsatz in den Anwendungsbereichen erfolgt eine gründliche *Evaluation* der resultierenden FMEA-Ontologie. Die Erfüllung der zuvor aufgestellten Anforderungen wird überprüft und möglicherweise wird die Anwendbarkeit der FMEA-Ontologie im späteren Systemumfeld getestet.

6.3.3 Einflüsse aus untersuchten Vorgehensmodellen

6.3.3.1 Einfluss aus dem Software Engineering

Mit dem OntoFMEA-Vorgehensmodell wird wie mit den Vorgehensmodellen des Software Engineering die ingenieursmäßige Entwicklung von Software (genauer: Ontologien) unterstützt. Die Rückkopplungen zwischen aufeinander folgenden Phasen, wie sie etwa mit dem Wasserfall-Modell vorliegen, wird im OntoFMEA-Vorgehensmodell ersetzt. Durch den Einsatz von schleifenförmigen Verzweigungen, die jedoch nicht einfach auf die vorhergehende

Aktivität verweisen, sondern auf noch weiter zurückliegende Aktivitäten, die bei einer Abweichung notwendig neu zu durchlaufen sind, werden für ausgewählte „Aktivitätenbündel" Rückkopplungen vorgesehen. Für die Phasen des Top-Levels wurden diese Schleifen insbesondere innerhalb der *Anforderungsspezifizierung* und der *Konzeptualisierung* berücksichtigt. Für die einzelnen Phasen des Top-Levels wird eine direkte Rückkopplung deshalb nicht für notwendig erachtet, weil die einzelnen Aktivitäten sorgfältig durchgeführt werden sollen, und weil es sich um einzelne abgeschlossene Aktivitäten handelt. Aufgrund ihrer überschaubaren Komplexität (im Gegensatz zu umfangreicheren Softwareentwicklungen wie bspw. der Entwicklung eines Betriebssystems) können diese Aktivitäten jeweils vollständig bearbeitet werden.

Die Idee des Spiral-Ansatzes für eine inkrementelle Software-Entwicklung wird vereinfacht berücksichtigt, indem die Phase der Evaluation bei einer unzureichenden Erfüllung von Anforderungen ein erneutes Durchführen der Phase der Wissensakqusition vorsieht. Hierzu wird unterstellt, dass sämtliche bis zu diesem Zeitpunkt erarbeiteten Ergebnisse bei einem erneuten Durchlauf weiter verwendet werden. Durch diese Schleife wird ein spiralförmiger Entwicklungsprozess, der eine kontinuierliche Verbesserung beinhaltet, umgesetzt.

Ein prototypischer Ansatz wird nicht berücksichtigt, weil der Verfasser der Ansicht ist, dass die Evaluation einer Ontologie erst erfolgen kann, wenn diese in einer quasi endgültigen Form vorliegt. Ansonsten ist davon auszugehen, dass bei etwaigen Schlussfolgerungen mit einer Inferenzmaschine auf der Grundlage einer unvollständigen Wissensbasis nicht eine aussagekräftige Evaluation hinsichtlich der zu Anfang der Entwicklung festgelegten Anforderungen möglich wird.

6.3.3.2 Einfluss aus dem Knowledge Engineering

Wie bereits erwähnt, wird das Ontology Engineering als eine Untergruppe des Knowledge Engineering angesehen. Als ein Vorgehensmodell zur Entwicklung von domänenspezifischen Ontologien beinhaltet das OntoFMEA-Vorgehensmodell als wichtiges Element des Modellbasierten Ansatzes des Knowledge Engineering die Erstellung eines *expliziten Modells* als Repräsentation von FMEA-Wissen für die Domäne des betrieblichen Qualitätsmanagements. Ein Schwerpunkt des OntoFMEA-Vorgehensmodells liegt wie bei sämtlichen Vorgehensmodellen des Knowledge Engineering in der Wissensakquisition, die deshalb auch als eine Phase im Top-Level Berücksichtigung findet.

6.3.3.3 Einfluss aus dem Ontology Engineering

Wie bei allen Vorgehensmodellen des Ontology Enginnering wird die informale Darstellung (Konzeptualisierung) des zu repräntierenden Wissens von der formalen Darstellung (explizites Modell) unterschieden. Das OntoFMEA-Vorgehensmodell wird in Kapitel 9.1, S. 279 ff., anhand der Anforderungen an Vorgehensmodelle des Ontology Engineering aus Kapitel 5.3.3.2, S. 170 ff., eingehend untersucht. Hieraus lassen sich im Umkehrschluss Erkenntnisse hinsichtlich des Einflusses der vorgestellten Ansätze des Ontology Engineering auf das OntoFMEA-Vorgehensmodell ziehen, die an genannter Stelle kurz angerissen werden.

6.4 Aufbau des OntoFMEA-Vorgehensmodells

Für die Darstellung des OntoFMEA-Vorgehensmodells wird auf die Modellierungsprimitive aus Kapitel 5.2, S. 162 ff., zurückgegriffen. Aus Redundanzgründen wird hier auf die erneute Darstellung der Modellierungsprimitive in Form einer Legende verzichtet.

Im Folgenden wird jeweils zuerst eine grafische (semi-formale) Darstellung der einzelnen Phase oder der phasenübergreifenden Unterstützungsleistung präsentiert. Im Anschluss wird die jeweilige Phase oder Unterstützungsleistung natürlichsprachlich beschrieben.

Eine schematische Gesamtübersicht des OntoFMEA-Vorgehensmodells findet sich zum Schluss dieses Kapitels 6 in Unterkapitel 6.4.3, S. 218. Aus drucktechnischen Gründen muss auf eine detaillierte Darstellung verzichtet werden. Die Gesamtdarstellung steht als MS-Visio-Zeichnung unter der URL: http://www.pim.uni-due.de/Dr_Lars_Dittmann.54.0.html zum Download bereit.

6.4.1 Phasenübergreifende Unterstützungsleistungen

6.4.1.1 Grafische Darstellung

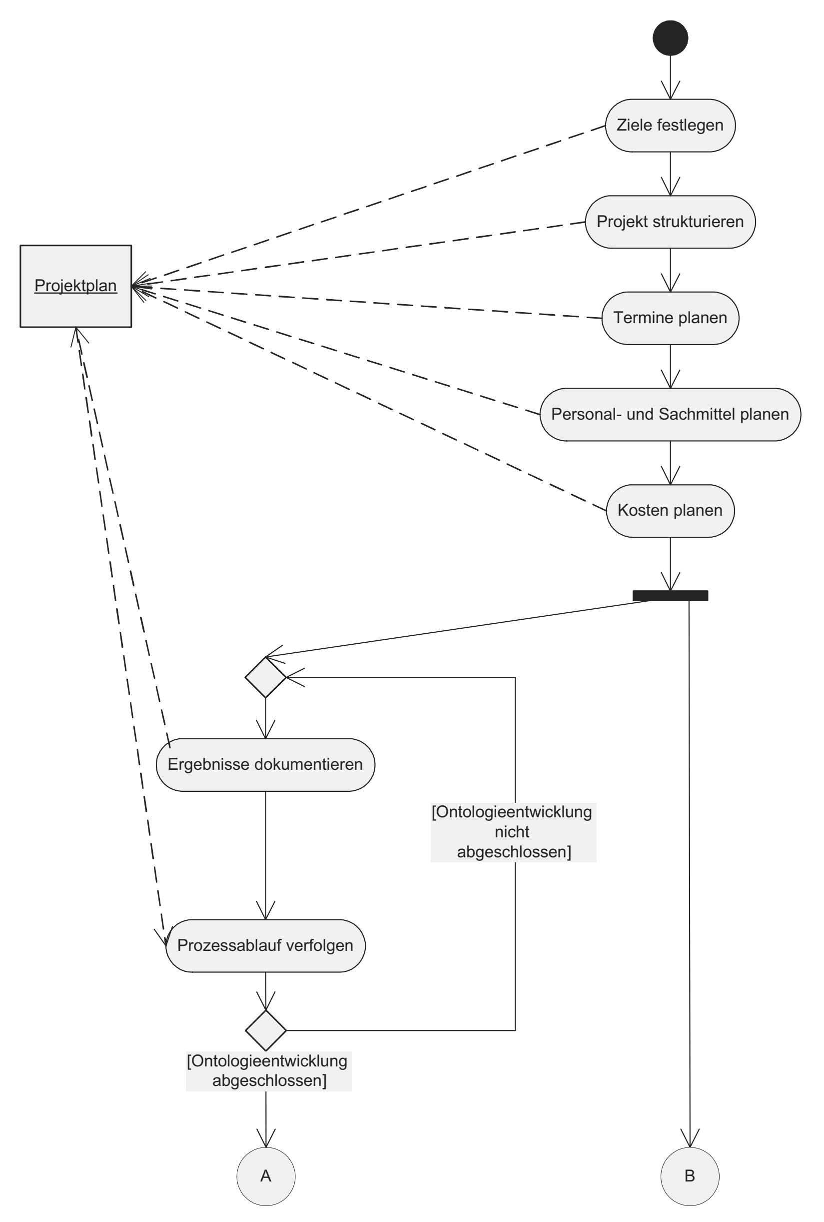


Abbildung 44: Phasenübergreifende Unterstützungsleistungen

6.4.1.2 Inhaltliche Beschreibung

Mit Beginn der FMEA-Ontologieentwicklung sind die Ziele, die mit der Entwicklung erreicht werden sollen, zu formulieren (*Ziele festlegen*). Die Ziele dienen als Handlungsorientierung während der Projektdurchführung.

Als zweiter Schritt findet sich die Aktion *Projekt strukturieren* in der Darstellung. In dieser Aktion wird der Projektaufbau inhaltlich und sachlogisch beschrieben.

Die zeitliche Ablauffolge des Projekts wird in der Aktion *Termine planen* festgelegt. Der Einsatz spezieller Softwareprodukte (bspw. MS Project) kann das Management der Ontologieentwicklung erheblich erleichtern und gleichzeitig wiederum die Dokumentation unterstützen, z. B. mit der automatischen Generierung von Reports.

Die Aktion *Personal- und Sachmittel planen* kann alternativ auch als Kapazitätsplanung bezeichnet werden. Während dieser Aktion werden die für die einzelnen Vorgänge vorhandenen sowie benötigten Personal- und Sachmittel festgestellt und abgeglichen. Anschließend werden Termine den einzelnen Vorgängen zugeordnet.

Die Kostenplanung (*Kosten planen*) schließt die vorbereitenden Aktionen der phasenübergreifenden Unterstützungsleistungen ab. In ihr werden die Kosten der Ontologieentwicklung anhand der übrigen Planungen verbindlich festgelegt.

Nach der Kostenplanung wird ein erster „imaginärer" Meilenstein erreicht.[771] Mit der vollständigen Planung wird entschieden, ob mit der Ontologieentwicklung weiter vorangeschritten wird. Es schließt sich im Modell eine Aufspaltung an, die zum Ziel hat, die verbleibenden Aktionen *Ergebnisse dokumentieren* und *Prozessablauf verfolgen* nebenläufig zur „eigentlichen" Ontologiekonstruktion (als „B" in der Abbildung zu finden) zu modellieren. Mittels einer Schleifenmodellierung wird sichergestellt, dass die beiden Aktionen so lange durchgeführt werden, bis die Ontologieentwicklung als abgeschlossen angesehen wird.

Für die jeweils nachfolgenden Aktivitäten im Ontologieentwicklungsprozess, aber auch für spätere Modifikationen oder Wiederverwendungen der Ontologie, ist eine gründliche Dokumentation von großer Bedeutung (*Ergebnisse dokumentieren*).[772] Aus diesem Grund soll das Projektteam parallel zur Ontologiekonstruktion eine genaue Beschreibung der relevanten Projektentscheidungen und -ergebnisse in textueller Form anfertigen. Im vorliegenden Modell werden alle Ergebnisse im Projektplan zentral abgelegt.

Unabhängig vom aktuellen Stand der Entwicklung müssen der Prozessablauf verfolgt und wichtige Ereignisse festgehalten werden (Aktion *Prozessablauf verfolgen*). Ein solches Ereignis ist bspw. das Erreichen eines Meilensteins im Projekt, etwa der (vorläufige) Abschluss ei-

771) Der Begriff „imaginär" deutet in diesem Fall darauf hin, dass dieser Meilenstein nicht explizit in der Modelldarstellung berücksichtigt wurde. Die verwendeten Modellierungsprimitive sehen eine solche Darstellung nicht vor.

772) Vgl. Fernández, Gómez-Pérez et al. (1997), S. 34.

ner Phase.[773] Die dabei als Ergebnisse entstandenen Artefakte (Anforderungsspezifikation, Konzeptualisierung, Ontologie usw.) bilden wiederum einen bedeutenden Teil der Dokumentation. Insbesondere die Verfolgung des Projektablaufs wird dem Projektmanagement zugerechnet.

Schwieriger, aber ebenfalls essentiell für die Verbesserung der Nachvollziehbarkeit und der Akzeptanz der FMEA-Ontologie ist die textuelle Fixierung wichtiger Entscheidungen[774] bezüglich des Vorgehens bei der Entwicklung. Während des gesamten Entwicklungsprozesses müssen die Mitglieder des Projektteams für diese Entscheidungen alle in Betracht gezogenen Alternativen, die letztendlich vorgenommene Auswahl und die dabei relevanten Gründe detailliert dokumentieren, um das Vorgehen für spätere Revisionen und für Rückfragen durch Personen, die nicht an dem Entwicklungsprozess beteiligt waren, transparent zu machen.

Die Dokumentation der Ontologieentwicklung ist erst abgeschlossen, wenn auch der Entwicklungsprozess selbst beendet ist. Auch in dieser phasenübergreifenden Unterstützungsleistung ist die Nutzung von Computerunterstützung sinnvoll. Der Einsatz bspw. von Software für kooperatives Arbeiten (Computer Supported Cooperative Work) oder von Microsoft-Office-Produkten verringert unter anderem den Koordinationsaufwand bei der Zusammenarbeit mehrerer Personen. Die meisten Ontologieentwicklungswerkzeuge bieten eine Hilfestellung für die Dokumentation, indem sie digitale, formale und oft auch grafische Darstellungen erleichtern und außerdem das Einfügen von Kommentaren und Erläuterungen im Quellkode erlauben.[775]

---

773) Siehe hierzu und zu den Ausführungen im folgenden Satz die Gesamtdarstellung des OntoFMEA-Vorgehensmodells (Kapitel 6.4.3, S. 218).

774) An dieser Stelle ist die Bedeutung von „wichtig“ kontextspezifisch und schwierig zu definieren; besondere Priorität sollte solchen Entscheidungen eingeräumt werden, die die Arbeit mehrerer Personen nachhaltig betreffen, damit bspw. im Nachhinein Veranwortlichkeiten berücksichtigt werden können.

775) So werden bspw. diese Hilfestellungen für die Dokumentation von den in Kapitel 4.1.4.3.1, S. 91 f., vorgestellten Entwicklungsumgebungen ermöglicht.

6.4.2 Phasen des Vorgehensmodells

6.4.2.1 Anforderungsspezifizierung

6.4.2.1.1 Grafische Darstellung

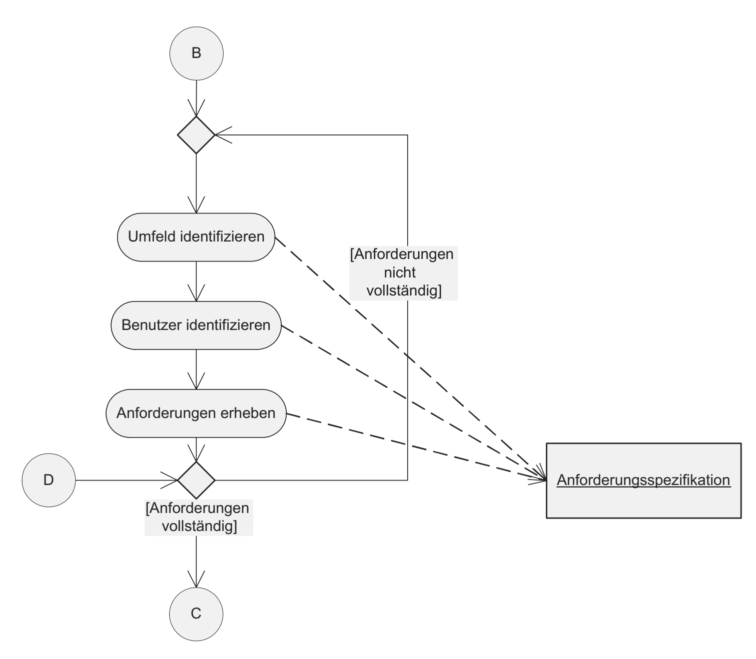


Abbildung 45: Phase Anforderungsspezifizierung

6.4.2.1.2 Inhaltliche Beschreibung

In der ersten Phase der Ontologieentwicklung, der *Anforderungsspezifizierung*, ist es wichtig, die Ziele der FMEA-Ontologie und ihre Anwendungsbereiche festzulegen, um möglichst alle Anforderungen zu erfassen, die während des Entwicklungsprozesses beachtet werden müssen. Neben dem Projektleiter und seinem Team sind dabei auch die späteren Benutzer der Ontologie und des Softwaresystems beteiligt. Ein wichtiges Ziel der Konstruktion von FMEA-Ontologien ist zum Beispiel eine effizientere Fehleranalyse. Nach der Spezifizierung des Hauptziels der Ontologieerstellung müssen diejenigen Anwendungsbereiche identifiziert werden, in denen die Ontologie zum Einsatz kommen soll (*Umfeld identifizieren*). Grundsätzliche Anwendungsbereiche sind bspw. die Abteilungen Konstruktion und Vertrieb, während einzelne Komponenten eventuell allen Beschäftigten zur Verfügung stehen sollen.

Sobald das technische und organisatorische Umfeld der Ontologie festgelegt ist (*Umfeld identifizieren*), können die zukünftigen *Benutzer identifiziert* und in Gruppen eingeteilt werden (zum Beispiel „Vertriebsmitarbeiter“, „Standardbenutzer – hauptsächlich lesender Zugriff“).

Anschließend könnte eine Befragung der Benutzer, gegebenenfalls der Repräsentanten einer Benutzergruppe, vorgenommen werden, um eine möglichst vollständige Anforderungsspezifikation erstellen zu können. Dafür muss das Projektteam eine Technik auswählen, bspw. strukturierte oder unstrukturierte Interviews, mittels derer die von den Benutzern gewünschten Anforderungen an die Ontologie erfasst werden.

Durch die Erstellung von Anwendungsfällen (Use Cases)[776] und Szenarien können die verschiedenen Situationen der Nutzung der Ontologie veranschaulicht und die spezifizierten Anforderungen ergänzt und verfeinert werden (*Anforderungen erheben*). Mögliche Anforderungen sind etwa „Die Ontologie muss erweiterbar sein." oder „Jeder Mitarbeiter soll nach einer Maschine mit einer bestimmten Funktion suchen können.".

Bei einer sehr großen und unübersichtlichen Menge von Anforderungen ist die Nutzung eines Anforderungsmanagement-Werkzeugs[777] sinnvoll, um eine computergestützte Verwaltung der Anforderungen, ihrer Abhängigkeiten und ihrer Änderungen zu erleichtern. Inzwischen ist auch ontologieentwicklungsspezifische Software verfügbar, die die Durchführung einer Anforderungsspezifizierung unterstützt.[778]

Die Identifizierung des Umfelds, die Befragung der Benutzer und die Analyse des Anwendungsbereichs werden so lange fortgeführt (Schleife im OntoFMEA-Vorgehensmodell), bis die Anforderungsspezifizierung von Entwicklern und Benutzern als vollständig angesehen wird. Allerdings können die Phasen auch „überlappend" umgesetzt werden, so dass das Projektteam nicht bis zum Abschluss der Anforderungsspezifizierung warten muss, bevor es mit der Wissensakquisition beginnt.

Die Abbildung 45 veranschaulicht den Ablauf der Phase der Anforderungsspezifizierung und das dabei entstehende Dokument (*Anforderungsspezifikation*).

---

776) Vgl. zum Beispiel Rumbaugh, Jacobson et al. (2005), S. 668. Anwendungsfälle sind ein Bestandteil der UML und umfassen in der Regel mehrere Szenarien, die durch ein gemeinsames Benutzerziel verbunden sind. Ein Szenario ist dabei eine mögliche konkrete Ausprägung eines Anwendungsfalls.

777) Beispiele für bekannte Anforderungsmanagementwerkzeuge sind etwa Requisite Pro von der Rational Software GmbH und DOORS von der Telelogic Deutschland GmbH.
Aktuelle Informationen können unter http://www-306.ibm.com/software/rational/ bzw. http://www.telelogic.com/products/doorsers/doors/index.cfm (Zugriffe am 16.02.2005) gefunden werden.

778) Zum Beispiel unterstützt OntoKick (ein Plug-in für die Werkzeugsammlung des On-To-Knowledge-Ansatzes) eine solche Durchführung (vgl. Sure, Studer (2002), S. 43 f.).

6.4.2.2 Wissensakquisition

6.4.2.2.1 Grafische Darstellung

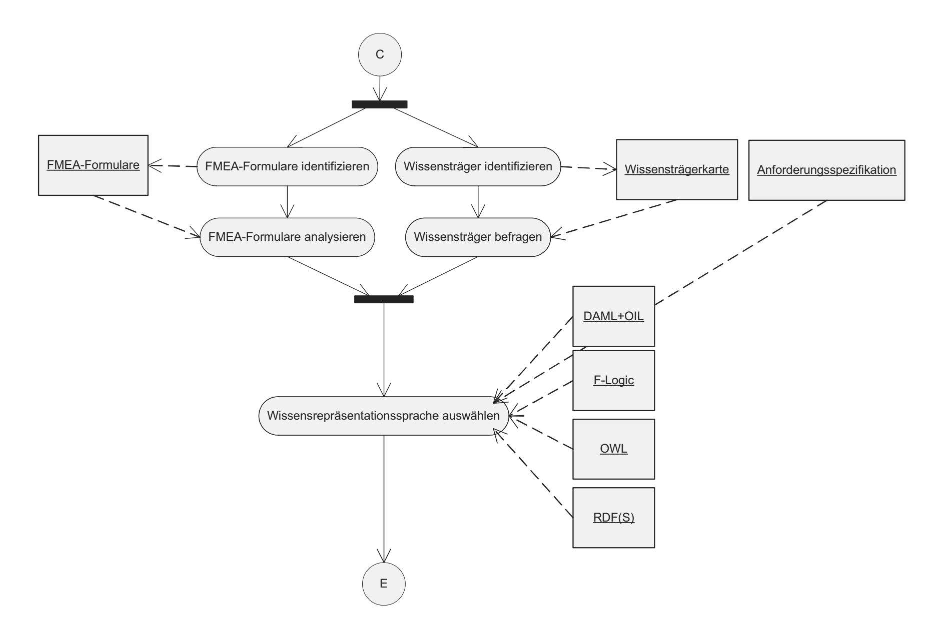


Abbildung 46: Phase Wissensakquisition

6.4.2.2.2 Inhaltliche Beschreibung

In der Phase *Wissensakquisition* muss das Projektteam das relevante Wissen für die Erstellung einer FMEA-Ontologie erfassen, das für ein erstes konzeptuelles Modell des Realitätsausschnitts der Ontologie benötigt wird. Hierzu sind sowohl die relevanten Wissensträger zu identifizieren und zu befragen als auch die relevanten FMEA-Formulare zu identifizieren und zu analysieren. Dies wird mittels eines nebenläufigen Prozessablaufs im Vorgehensmodell dargestellt.

Zum einen sollen diejenigen *Mitarbeiter identifiziert* werden, die für die FMEA-Durchführung im Unternehmen allgemein (meist Mitarbeiter der Konstruktionsabteilung) oder in den einzelnen Organisationseinheiten (so genannte Domänenexperten oder auch die Abteilungsleiter) verantwortlich sind. In einer *Wissensträgerkarte* wird die Liste relevanter Mitarbeiter vorgehalten. Durch eine *Befragung* dieser Mitarbeiter werden Informationen über betrachtete Systeme und implizite Annahmen, die nicht in den FMEA-Formularen expliziert wurden, sowie über die „vorherrschende Begriffswelt“ im Unternehmen und über eventuell existierende weitere Dokumente mit Wissen über FMEA-Anwendungen erhoben.

Dabei ist zu beachten, dass es hier nicht allein um das Aufnehmen von Wissen, sondern von Wissen über Wissen (welches in den Formularen abgelegt wurde) geht und damit teilweise die Ebene des Meta-Wissens im Vordergrund steht. Diese Besonderheit kann einen erhöhten Aufwand gerade bei der Wissensakquisition mit sich bringen, da Wissen über oftmals nur implizit vorhandenes Wissen, wie z. B. Konstruktionsannahmen, schwer zu formulieren und zu erfassen ist. Der Prozess der Explizierung impliziten Wissens, von Nonaka und Takeuchi als „Externalisierung" bezeichnet[779], ist essentiell für die Generierung neuen Wissens und kann durch Techniken wie Metaphern und Analogien unterstützt werden.[780] Wenn *Meta*-Wissen „externalisiert" werden soll, ist auch die Schaffung eines Bewusstseins für das Vorhandensein und die Relevanz dieses Wissens von Bedeutung, weil die Externalisierung hauptsächlich über die Mitarbeiter und deren Bereitschaft (Motivation) zur Explikation erfolgen muss.

Zusätzlich zu den Mitarbeiterbefragungen durch Interviews oder auch Brainstorming-Sitzungen werden alle bereits erstellten *FMEA-Formulare* vom Projektteam *identifiziert*, aufgelistet und anschließend hinsichtlich der enthaltenen Konzepte und Relationen *analysiert* (siehe Abbildung 46). Die durch die Experteninterviews und Textanalysen identifizierten Konzepte und Relationen bilden den Ausgangspunkt für die Basis-Terminologie, die dann schrittweise durch weitere Wissensakquisition verfeinert und ergänzt wird.

Anschließend wird eine Entscheidung hinsichtlich der *Auswahl einer Wissensrepräsentationssprache* getroffen. Hierzu wird auf der Basis der Anforderungsspezifikation eine geeignete Wissensrepräsentationssprache für die Ontologie ausgewählt (für das OntoFMEA-Projekt sind die nahe liegenden Alternativen – unter anderem wegen der zur Verfügung stehenden Computer-Werkzeuge – zum Beispiel F-Logic, RDF(S), DAML+OIL und OWL). Der Grund für die frühzeitige Festlegung auf eine Wissensrepräsentationssprache ergibt sich aus der Erkenntnis, dass die Modellierungsprimitive der Repräsentationssprache (Meta-Meta-Ebene) sich auf die Modellierung der Konzepte und Relationen der FMEA-Ontologie (Meta-Ebene und Klassen-Ebene) auswirken.[781] Um unnötigen Mehraufwand durch Revision von Konzepten und Relationen zu vermeiden, weil bspw. bestimmte Relationen in einer Sprache überhaupt nicht darstellbar sind (etwa drei- und mehrstellige Relationen), legt man sich bereits frühzeitig auf eine Sprache fest. Die weitere Ontologieentwicklung innerhalb der Konzeptualisierung erfolgt somit „in den Gedanken in der Sprache" der Repräsentation, d.h. bspw., falls eine Repräsentationssprache lediglich zweistellige Relationen erlaubt, so sollten in der Konzeptualisierung nur solche Relationen berücksichtigt werden, um einen langwierigen Transformationsprozess zu vermeiden. Weil diese „neue" Sprache als Objekt selbst (bspw. ihre Modellierungsprimitive) eine Erweiterung des Wissens der Projektteammitglieder darstellt (bspw. das Wissen über die mögliche Stelligkeit von Relationen), wird sie in der Phase der Wissensakquisition verankert.

779) Vgl. Nonaka, Takeuchi (1997), S. 75 ff.

780) Vgl. Weiss, Menzel et al. (2005), S. 130.

781) Siehe hierzu auch die Ausführungen in Kapitel 6.2.3, S. 196 ff.

6.4.2.3 Konzeptualisierung

6.4.2.3.1 Grafische Darstellung

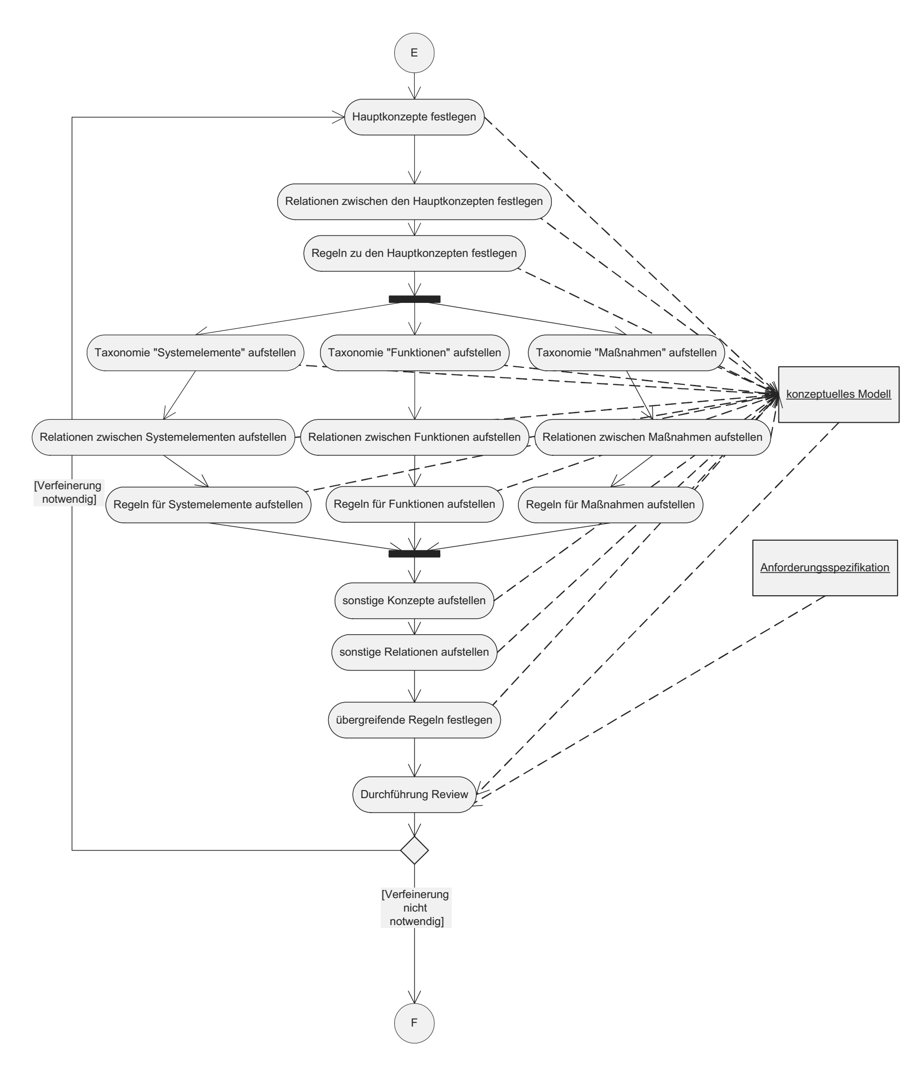


Abbildung 47: Phase Konzeptualisierung

6.4.2.3.2 Inhaltliche Beschreibung

Bei der *Konzeptualisierung* wird ein externes konzeptuelles Modell eines Realitätsausschnitts erarbeitet, das einerseits ein Konzeptsystem für die Domäne in Form einer Terminologie und andererseits Regeln für die semantisch korrekte Verwendung der Konzepte beinhaltet. Beteiligt sind dabei nicht nur die Mitglieder des Projektteams, sondern auch die schon bei der Wissensakquisition befragten Domänenexperten, damit eine realitätsnahe Ontologie aufgebaut werden kann.

Um die Nachvollziehbarkeit der Entwicklung zu gewährleisten, ist eine durchgängige Dokumentation dieser Phase, insbesondere der konkreten Vorgehensweise und der getroffenen Entscheidungen, von besonderer Bedeutung.

Für die Durchführung der Konzeptualisierung wird in der Literatur meist ein so genannter *Middle-Out-Ansatz* empfohlen.[782] Bei diesem Ansatz werden, ausgehend von den relevantesten Konzepten (bspw. die am häufigsten genannten Konzepte), zunächst domänen- oder abteilungsspezifische Terminologie-„Inseln“ erstellt, aus denen später die gesamte Konzeptualisierung gebildet wird. Eine andere Möglichkeit ist der *Top-Down-Ansatz*. Hierbei sind zuerst die allgemeinsten Konzepte als oberste Ebene für die Konzeptualisierung zu identifizieren, damit diese anschließend auf untergeordneten Ebenen spezialisiert werden können. Die Anwendung dieser Methode kann sehr tief strukturierte, umfassende Ontologien hervorbringen, doch sie setzt die Existenz entsprechender Informationen und Schemata zu Umfang und Reichweite der Domäne voraus. Zusätzlich braucht der Entwickler Erfahrung im Umgang mit konzeptueller Modellierung und Ontologien.

Die FMEA-Formblätter werden bereits als umfassendes und tief strukturiertes Wissen über eine Domäne angesehen.[783] Deshalb wird der Top-Down-Ansatz im Modell angewendet. Die Ontologieentwickler erstellen in Zusammenarbeit mit Domänenexperten eine Top-Level-Konzeptualisierung, indem sie die Konzepte auf der obersten Abstraktionsebene identifizieren und damit erste Klassen zur Spezialisierung vorgeben. Dies sind die *Hauptkonzepte* der FMEA-Ontologie. Zu den Hauptkonzepten werden *Relationen* und *Regeln* festgelegt.

---

782) Dies ist auch in den meisten der in Kapitel 5.1.2.3 dargestellten Vorgehensmodelle der Fall; vgl. Uschold, King (1995), S. 9 f.; Grüninger, Fox (1995), S. 5; Fernández, Gómez-Pérez et al. (1997), S. 36 f.; Sure, Studer (2002), S. 48.
Falls das Wissen der Entwickler noch begrenzt und unstrukturiert ist, sollte die Ontologiekonstruktion nach dem *Middle-Out-Ansatz* durchgeführt werden. Ontologieentwickler und Abteilungsrepräsentanten identifizieren die relevantesten Konzepte und beschreiben diese durch Bezeichner, Attribute und Relationen und eventuell Integritätsregeln. Dann ergänzen sie die Konzepte in Gruppenarbeit (zum Beispiel für jede Organisationseinheit) und ordnen sie hierarchisch. Auf diese Art und Weise werden in jedem Bereich verschiedene Terminologie-„Inseln“ mit zusammenhängenden Begriffen gebildet, die dann miteinander zu verknüpfen sind. Die Zusammenführung verursacht oft einen hohen Aufwand, da bei der Verbindung leicht Redundanzen und verwirrende Strukturen entstehen (vgl. Lau, Sure (2002), S. 129); doch sowohl Redundanzen als auch Verwirrungen können zu einer ersten „ad hoc“-Evaluation genutzt werden.

783) Siehe hierzu die Ausführungen in Kapitel 3.3, S. 62 ff.

Ausgehend von dieser Grundstruktur wird die Ontologie anschließend erweitert und verfeinert. Es werden *Taxonomien* zu *Systemelementen*, *Funktionen* und *Maßnahmen* festgelegt.[784] Die Terminologieverfeinerung wird fortgesetzt mit der Modellierung von *Relationen* zu den Konzepten der Taxonomien. Sukzessive zu der Terminologieverfeinerung formulieren die Ontologieentwickler die semantischen *Regeln*, die einerseits das implizit enthaltene Wissen als explizite Schlussfolgerungen erschließen (Inferenzregeln), aber auch in Form von Integritätsregeln die Zulässigkeit von Verknüpfungen der definierten Konzepte einschränken.

Nach der Entwicklung der Taxonomien mit zusätzlichen Relationen und Regeln folgen drei weitere Aktionen, in deren Verlauf die „*sonstigen*" *Konzepte*, *Relationen* und *Regeln* aufgestellt werden sollen. Hierzu gehören bspw. die Konzepte zu möglichen Fehlern. An dieser Stelle wird auch besonders an eine Integrationsbemühung gedacht, d. h., es soll nach Konzepten, Relationen und Regeln gesucht werden, die die drei isolierten Taxonomien mit den Hauptkonzepten zu einer Einheit verbinden.

Das Vorgehen bei der Definition von Regeln ist schwierig und wegen der unterschiedlichen Strukturen und Zusammenhänge einzelner Ontologien kaum systematisierbar. Die Entwickler sollten darauf achten, dass die Menge aller Regeln ausreichend ist, um alle Anforderungen an die Aussagekraft der Ontologie zu erfüllen.[785] Die Anforderungen hinsichtlich der Aussagekraft lassen sich mit den bereits erwähnten Kompetenzfragen formulieren.[786]

Die Terminologie und die verschiedenen Inferenz- und Integritätsregeln der späteren Ontologie sind zu diesem Zeitpunkt noch informal (oder eventuell semi-formal) durch textuelle und grafische Repräsentationsarten dargestellt und bilden damit die *Konzeptualisierung*.

Bevor dieses konzeptuelle Modell der Domäne in eine formale Spezifikation des Wissens in der Implementierungsphase transformiert wird, soll das Ergebnis der bisherigen Ontologieentwicklung beurteilt werden, um eventuelle Fehler möglichst frühzeitig aufzudecken. Daher sollte das Projektteam durch die *Durchführung* von *Reviews* (im Modell als Schleife berücksichtigt mit einer *Verzweigung*; Wächterbedingungen der Verzweigung entscheiden, ob eine Verfeinerung notwendig ist oder nicht), an denen auch Vertreter der Benutzer teilnehmen können, die Qualität der Konzeptualisierung überprüfen. Eventuell an die Ontologierepräsentation gestellte Anforderungen der Benutzer und Entwickler sind ebenso zu beachten wie generelle Design-Kriterien, etwa Klarheit (die bspw. durch Vollständigkeit erreicht werden kann), Kohärenz (bspw. müssen die Inferenzregeln und Integritätsregeln sowohl untereinan-

784) Im Laufe der Untersuchungen ist der Verfasser zu dem Schluss gekommen, dass es nicht notwendig erscheint, extra eine Taxonomie mit möglichen Fehlern vorzuhalten, weil diese nicht sinnvoll handhabbar wäre (aufgrund der beliebig vergrößerbaren Zahl denkmöglicher Fehler) und in der Regel die Fehler direkt aus den Funktionen abgeleitet (zumeist über eine einfache Negation) werden können. Dennoch müssen zu einem späteren Zeitpunkt Konzepte zu Fehlern in der Ontologie repräsentiert werden.
Ein Beispiel für eine mögliche „feinere" Untergliederung der Aktivität *Taxonomie „Funktionen" aufstellen* wäre bspw. möglich durch eine Erweiterung um die Aktivität *Funktionsstruktur erstellen*. Hierzu liefert Puri ein Aktivitätsdiagramm (vgl. Puri (2003), S. 93). Alternativ nennen Kitamura und Mizoguchi einen ontologiebasierten Modellierungsprozess für die Erstellung einer Funktionsstruktur, der nicht als UML-Aktivitätsdiagramm vorliegt (vgl. Kitamura, Mizoguchi (2004), S. 104 ff.).

785) Vgl. Grüninger, Fox (1995), S. 7.

786) Siehe hierzu die Ausführungen zum TOVE-Ansatz, Kapitel 5.1.2.3.1.3, S. 139 ff.

der als auch in Bezug auf die Begriffsdefinitionen kohärent sein), Erweiterbarkeit und minimale ontologische Bindung (es sind möglichst wenige Forderungen an den modellierten Realitätsausschnitt zu stellen).[787] Für die Umsetzung der Reviews ist bspw. die Delphi-Methode geeignet[788], bei der die Konzeptualisierung iterativ immer wieder modifiziert und verbessert wird, bis sie aus Sicht aller Beteiligten als vollständig anzusehen ist und den Anforderungen entspricht.

Sobald zwischen allen Teilnehmern der Reviews eine Übereinstimmung hinsichtlich des Inhalts und des Designs der Konzeptualisierung erzielt wird, kann mit der Implementierung begonnen werden.

787) Vgl. Gruber (1993), S. 200 f. Die exemplarisch genannten Anforderungen an die Ontologierepräsentation ähneln den Anforderungen aus Kapitel 5.3.3.2, S. 170 ff., an Vorgehensmodelle des Ontology Engineering; sie sind jedoch nicht identisch, weil sie unterschiedliche Bezugsobjekte adressieren.

788) Vgl. Holsapple, Joshi (2002), S. 45 ff. Ein Grund für die Eignung der Delphi-Methode liegt bspw. in der gemeinsamen Anwendung der Methode durch ihre Benutzer. Hierdurch kann die gemeinschaftliche Konzeptualisierung auch gemeinschaftlich überprüft werden.

6.4.2.4 Implementierung

6.4.2.4.1 Grafische Darstellung

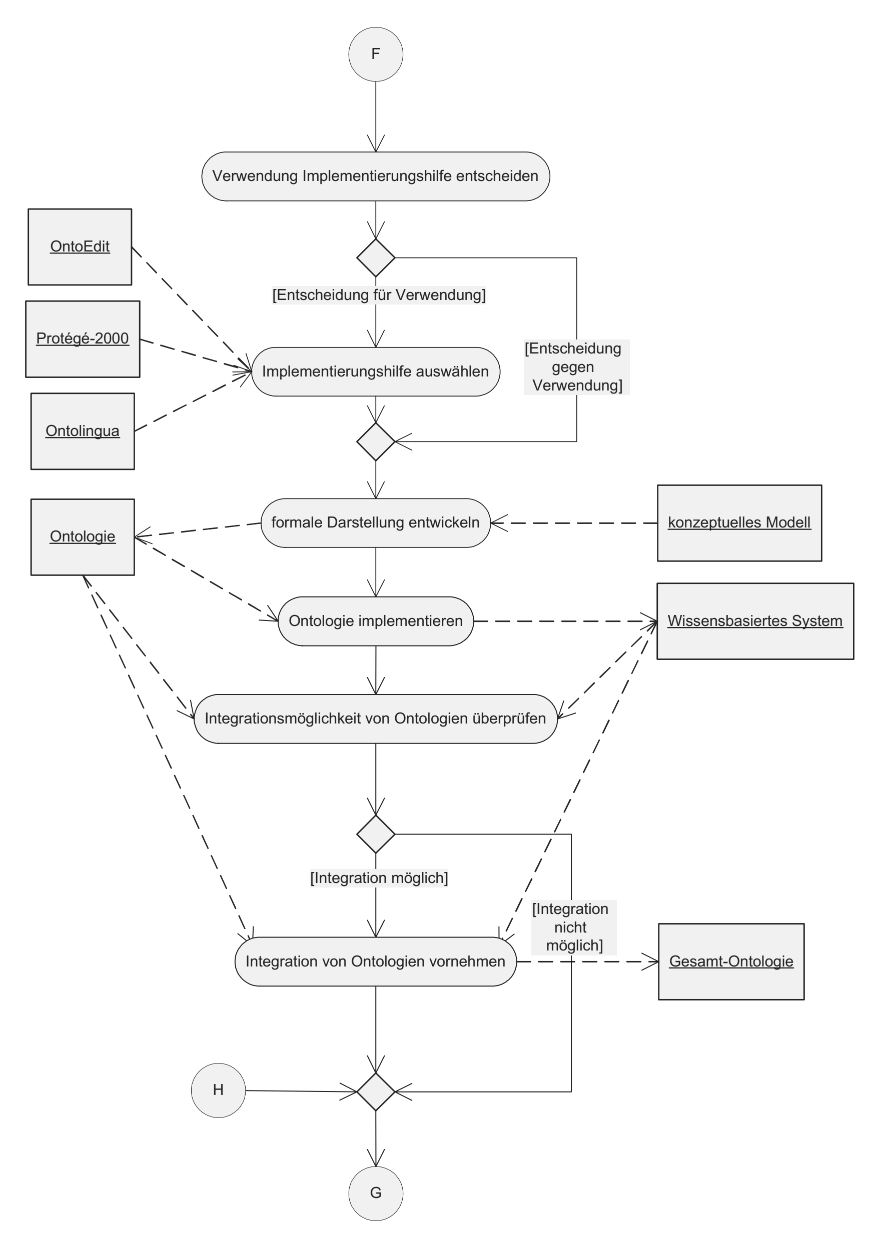


Abbildung 48: Phase Implementierung

6.4.2.4.2 Inhaltliche Beschreibung

Ähnlich wie die Phase Kodierung des Enterprise-Model-Ansatzes[789] umfasst die Implementierung die Erstellung einer formalen Repräsentation (Spezifikation) des Domänenwissens des konzeptuellen Modells, das am Ende der Konzeptualisierungsphase vorliegt, und die Integration in ein lauffähiges System. Die formale Repräsentation des Domänenwissens soll getrennt von der Konzeptualisierung durchgeführt werden, weil die Konzeptualisierung so wenig wie möglich auf bestimmte formale, computergestützt verarbeitbare Sprachen oder andere technische Anforderungen ausgerichtet sein soll.

Die Ontologieentwickler haben bereits eine formale Repräsentationssprache ausgewählt. Dabei mussten sie auf eventuelle Benutzeranforderungen bezüglich der Funktionalität der Ontologie sowie auf Systemumgebung und gegebene Nebenbedingungen Rücksicht nehmen.

Eine Implementierungshilfe kann die Aktivitäten der Formalisierung erleichtern und sogar zu einem großen Teil automatisieren, schränkt jedoch auch die Anzahl der zur Verfügung stehenden formalen Sprachen ein.[790] Es ist deshalb zu entscheiden, ob eine *Implementierungshilfe ausgewählt* wird. Die Erfassung und Verwaltung des Domänenwissens[791] kann durch die Nutzung computergestützter Werkzeuge als Implementierungshilfe erleichtert werden. Um die Entwicklung, konzeptuelle Modellierung und Dokumentation der FMEA-Ontologie zu unterstützen, können Software-Werkzeuge wie OntoEdit[792], Protégé-2000[793], Entwicklungsumgebung Ontolingua[794] oder auch KAON[795], ein Programm für die Konstruktion, Formalisierung und Verwaltung von Ontologien, eingesetzt werden.[796]

---

789) Vgl. Uschold, King (1995), S. 3.

790) Eine solche Entwicklungsumgebung kann möglicherweise schon während der Konzeptualisierung genutzt werden. Der Verfasser schlägt jedoch, um Kodierungsverzerrungen zu vermeiden, vor, solche Hilfsmittel so spät wie möglich einzusetzen.

791) Als Domäne gilt hier das Wissen, das in einem FMEA-Formular enthalten ist.

792) OntoEdit ist der Kern der On-To-Knowledge-Werkzeugsammlung der Ontoprise GmbH. Es stellt Funktionalitäten für die Konstruktion, insbesondere für die Darstellung von Ontologien bereit und unterstützt die Ausgabesprachen XML, F-Logic, RDF(S) und DAML+OIL. Vgl. Sure, Studer (2002), S. 21 und 50 f.

793) Protégé-2000 wurde von der Stanford Medical Informatics Group (SMI) an der Stanford Universität entwickelt; aktuelle Informationen sind unter http://protege.stanford.edu/, Zugriff am 18.03.2007, zu erhalten. Protégé-2000 beinhaltet u. a. Funktionalitäten zur Visualisierung und Bearbeitung von Ontologien in grafischer Form und ermöglicht bspw. die Implementierung in F-Logic, OIL und RDF(S).

794) Vgl. Farquhar, Fikes et al. (1996), S. 44.3 ff. Durch Ontolingua können Ontologien in die gleichnamige Sprache Ontolingua, aber auch in Prolog und LOOM und CLIPS übersetzt werden (siehe hierzu auch Kapitel 4.2.2.1.2, S. 104 ff.).

795) KAON (Karlsruhe Ontology and Semantic Web Tool Suite) kann zur Entwicklung ontologiebasierter Anwendungen genutzt werden. Es wurde, wie auch die On-To-Knowledge-Werkzeuge, am Institut AIFB (Angewandte Informatik und Formale Beschreibungsverfahren) der Universität Karlsruhe entwickelt, stellt jedoch im Gegensatz zur On-To-Knowledge-Werkzeugsammlung eine nicht-kommerzielle Werkzeugsammlung dar (vgl. http://kaon.semanticweb.org/, Zugriff am 18.03.2007).

796) Siehe hierzu die Fn. 790.

Nach der optionalen Auswahl einer Implementierungshilfe transformiert das Projektteam das konzeptuelle Modell in eine formale Darstellung (*formale Darstellung entwickeln*). Dabei sollen die an die Spezifikation gestellten Anforderungen der Benutzer und Entwickler ebenso beachtet werden wie die bereits kurz genannten generellen Design-Anforderungen an eine Ontologie, insbesondere Klarheit, Objektivität, Formalität, Kohärenz, Erweiterbarkeit und minimale Verzerrung durch die Kodierung.

Nachdem durch die Transformation die Ontologie erstellt wurde, kann das Projektteam die Ontologie im Rahmen einer Software implementieren, damit die Ontologie in den Anwendungsbereichen auch computergestützt genutzt werden kann. Diese Software dient bspw. der Realisierung eines ontologiebasierten FMEA-Werkzeugs. Aus diesem Grund gehört dazu unter anderem die Entwicklung von Oberflächen für die Benutzerinteraktion (bspw. für die Visualisierung und Verwaltung von bereits durchgeführten FMEAs). Ebenso gehört dazu die Programmierung einer Überwachung der Integritäts- und Inferenzregeln, etwa in Form einer Inferenzmaschine (bspw. kann die Software Vorschläge hinsichtlich möglicher Fehler eines zu untersuchenden Systems bei der Durchführung einer FMEA unterbreiten). Falls bereits ein solches System auf der Basis von Ontologien existiert und darüber hinaus auch Ontologien hierzu existieren, ist auch eine *Integration* der neuen FMEA-Ontologie mit den sich bereits im Einsatz befindlichen Ontologien *vorzunehmen* (auch Ontologie-Mapping genannt[797]).

6.4.2.5 Evaluation

6.4.2.5.1 Grafische Darstellung

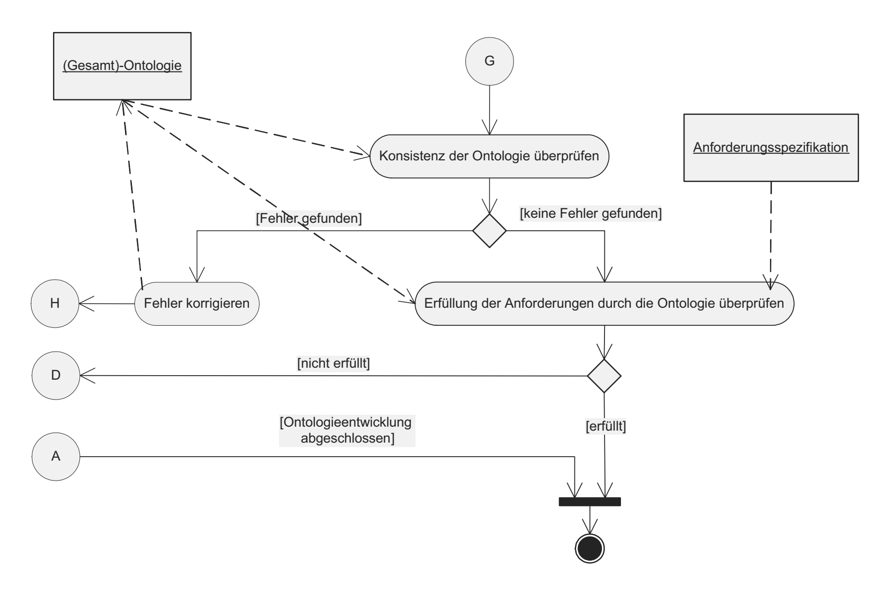


Abbildung 49: Phase Evaluation

797) Zum Ontology Mapping siehe auch die Untersuchung von Kalfoglou, Schorlemmer (2003), S. 1 ff.

6.4.2.5.2 Inhaltliche Beschreibung

Bevor die FMEA-Ontologie eingeführt und benutzt werden kann, muss sie im Hinblick auf die Erfüllung der Benutzeranforderungen und ihre generelle Anwendbarkeit im IT-Systemumfeld bewertet werden (*Evaluation*). In dieser Phase wird daher eine Evaluation der Ontologie vorgenommen, d. h. eine Beurteilung ihrer Funktionalitäten hinsichtlich eines Bezugsrahmens[798], den die Anforderungsspezifikation mit ihren funktionalen und nicht funktionalen Anforderungen bildet.

Die Aktion *Konsistenz der Ontologie prüfen* untersucht die Frage, ob die Ontologie formal korrekt aufgebaut wurde und damit im Sinne der formalen nicht-funktionalen Anforderungsspezifikation korrekt ist, während bei der Aktion *Erfüllung der Anforderungen durch die Ontologie überprüfen* geprüft wird, ob das Programm in einer bestimmten Zielumgebung lauffähig ist und insbesondere die vom Benutzer gewünschten (funktionalen) Anforderungen liefert.

Indem die Entwickler die *Konsistenz der Ontologie prüfen*, analysieren die Entwickler die einzelnen Bestandteile der Ontologie und untersuchen dabei, ob alle Konzepte, Relationen und Regeln korrekt definiert sind. Außerdem werden die Aussagen aller Integritäts- und Inferenzregeln auf ihre eigene Konsistenz und auf ihre Konsistenz untereinander sowie zu den ermittelten Konzepten und Relationen geprüft. Festgestellte Mängel werden direkt korrigiert und dokumentiert.

Bei der Aktion *Erfüllung der Anforderungen durch die Ontologie überprüfen* beginnt das Entwicklerteam, alle Konzepte, Relationen und Regeln auf ihre Erfüllung der Benutzer- und der Entwickleranforderungen gemäß der Anforderungsspezifikation zu analysieren. Zudem bewerten die Entwickler zusammen mit Vertretern der Benutzer, ob die Ontologie tatsächlich die erforderliche Repräsentation für den betreffenden Realitätsausschnitt bereitstellt, den sie repräsentieren soll. Von besonderer Bedeutung ist der Vergleich mit dem im Rahmen der funktionalen Anforderungsspezifizierung näher beschriebenen Einsatzzweck „Repräsentation von FMEA-Wissen“. Die fertiggestellte FMEA-Ontologie muss die eingangs festgelegten Anforderungen erfüllen und dafür diejenigen Leistungen erbringen, die in den Benutzeranforderungen als zu implementierende Funktionalitäten definiert werden.

Es ist wichtig, dass die Anwendbarkeit der Ontologie in ihren späteren Anwendungsbereichen gewährleistet ist. Daher sollte, sobald alle Komponenten des gesamten ontologiebasierten FMEA-Systems implementiert sind, die FMEA-Ontologie (mit ihrer Wissensbasis) als Teil dieses Systems in den Anwendungsbereichen getestet werden. Durch eine Simulation der tatsächlichen Nutzung im Unternehmen können die Entwickler die Erfüllung der technischen Anforderungen, etwa die Interoperabilität und Kooperation mit anderen Systemen wie z. B. Konstruktionsprogrammen, überprüfen. Auf der anderen Seite spielt auch die Benutzerfreundlichkeit der Bedienung, die unter anderem durch die Performanz und die Zugangsmöglichkeiten der verschiedenen Benutzer beeinflusst wird, eine große Rolle, weil der Nutzen der Ontologie von der Akzeptanz der Benutzer abhängt. Die Aktion des Testens in den Anwendungs-

798) Vgl. Gómez-Pérez (1994), S. 11.

bereichen wird als Bestandteil der Aktivität *Erfüllung der Anforderungen durch die Ontologie überprüfen* angesehen, ihre Durchführbarkeit ist jedoch abhängig von der Implementierung des gesamten FMEA-Systems, dessen Fertigstellung nicht mit dem Abschluss der Ontologieimplementierung zusammenfallen muss. Wenn schwerwiegende inhaltliche Probleme vorliegen, muss das gesamte konzeptuelle Modell überarbeitet werden (und auch alle der Konzeptualisierung nachfolgenden Phasen müssen noch einmal durchlaufen werden). Hierzu kann es notwendig werden, eine erneute Wissensakquisition zu durchlaufen: Es wurde im Modell eine Verzweigung mit der Wächterbedingung „nicht erfüllt" modelliert. Falls die Anforderungen nicht erfüllt werden, die durch die Anforderungsspezifizierung dokumentiert wurden, so verzweigt das OntoFMEA-Vorgehensmodell an den Anfang der Wissensakquisition.[799] Je nach Grad der Erfüllung der Anforderungen werden die nachfolgenden Phasen aktiv oder passiv durchlaufen. Sollte eine erneute Wissensakquisition als nicht notwendig erachtet werden, so wird dieser Teil passiv, d. h. die Aktionen werden als bereits erfüllt angesehen, ohne eine erneute Aktionsdurchführung zu durchlaufen.

Falls alle Beteiligten darin übereinstimmen, dass die Ontologie der Anforderungsspezifikation gerecht wird, wird die FMEA-Ontologieentwicklung als abgeschlossen angesehen. Der Endzustand im OntoFMEA-Vorgehensmodell wird erreicht.

Wie auch in den anderen Phasen der FMEA-Ontologieentwicklung ist bei der Evaluation die Nutzung von Software-Werkzeugen möglich, um das Vorgehen zu erleichtern, möglicherweise (teilweise) zu automatisieren und den Überblick zu behalten. Bisher existieren jedoch (auch in der Literatur) nur wenige explizit für die Ontologieevaluation entwickelte Programme, bspw. OCM[800] sowie OntoAnalyser und OntoGenerator[801].

---

799) Der Konnektor „D" führt im Effekt direkt zur Phase *Wissensakquisition.* Lediglich aus modelltechnischen Gründen (in eine Aktion soll lediglich eine Kontrollflusskante münden) wird noch in das Ende der Phase *Anforderungsspezifizierung* bei der geteilten Darstellung der einzelnen Phasen verzweigt.

800) Der „Ontological Constraints Manager" (OCM) wurde ursprünglich zur Konsistenzprüfung für die Verbesserung der Systemzuverlässigkeit entwickelt; das Werkzeug hat jedoch vielfältige Anwendungsmöglichkeiten und kann auch für die Evaluation von Ontologien selbst eingesetzt werden. Vgl. Kalfoglou, Robertson et al. (1999), S. 13 ff.

801) OntoAnalyser und OntoGenerator sind zwei Plug-Ins für das bereits erwähnte Werkzeug „OntoEdit"; hier ist besonders OntoAnalyser von Bedeutung, da das Programm zur Überprüfung von Ontologieeigenschaften dient (z. B. sprachliche Konformität und Konsistenz), während OntoGenerator für die Evaluation ontologiebasierter Anwendungen konstruiert wurde. Vgl. dazu Angele, Sure (2002), S. 3 ff.

6.4.3 Schematische Gesamtdarstellung OntoFMEA-Vorgehensmodell

Abbildung 50: Schematische Gesamtdarstellung OntoFMEA-Vorgehensmodell

Die detaillierte Gesamtdarstellung des OntoFMEA-Vorgehensmodells kann als MS-Visio-Zeichnung unter der URL: http://www.pim.uni-due.de/Dr_Lars_Dittmann.54.0.html abgerufen werden.

7 FMEA-Ontologie

7.1 Grundlagen der FMEA-Ontologie

Das folgende Kapitel vertieft die bisherigen Ausführungen zur Entwicklung einer FMEA-Ontologie. Es soll den Einsatz des OntoFMEA-Vorgehensmodells zur Entwicklung einer FMEA-Ontologie verdeutlichen. Zu diesem Zweck werden die Komponenten der FMEA-Ontologie detailliert vorgestellt, und es wird jeweils erläutert, welche Designentscheidungen die Gestaltung der Ontologiekomponenten maßgeblich geprägt haben.

Der gegenwärtige Stand der Forschung im Bereich der computergestützten FMEA-Anwendung innerhalb des Qualitätsmanagements besteht darin, FMEA-relevantes Wissen in hierarchischen Strukturen zu repräsentieren.[802] Diese werden oftmals als Struktur-, Funktions- und Fehlfunktionsbäume dem Benutzer zur Verfügung gestellt.

Die Abbildung 51 zeigt exemplarisch eine Visualisierung der FMEA-Struktur technischer Zusammenhänge, nachdem eine technische Analyse eines Maschinensystems durchgeführt wurde.

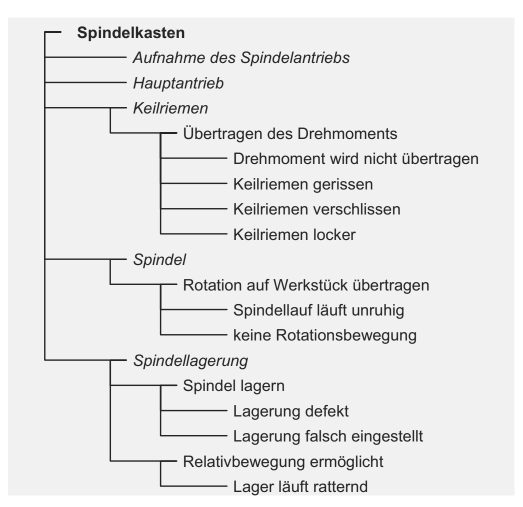


Abbildung 51: Ausschnitt integrierter Struktur-, Funktions- und Fehlfunktionsbaum[803]

Ferner verdeutlicht die Abbildung exemplarisch den aktuellen Stand der Forschung in diesem Bereich, weil Ontologien für die Zwecke des fehlerbezogenen Qualitätsmanagements, die konkret das Instrument FMEA berücksichtigen, bis auf einen rudimentären, „oberflächlichen" Ansatz von Lee[804] noch nicht existieren. Ontologien für die Zwecke des betrieblichen Quali-

802) Zu einer ausführlichen Darstellung des aktuellen Einsatzes der (computergestützten) FMEA siehe Kapitel 3, S. 35 ff., insbesondere die Unterkapitel 3.1.4.2, S. 52 ff., und 3.1.4.3, S. 54 ff.

803) Vgl. Timpe, Seibicke (2000), S. 1433.

804) Siehe hierzu Kapitel 2.2, Seite 34.

tätsmanagements sind jedoch wünschenswert, weil sie auf die Wiederverwendung von Wissen und Computerverarbeitbarkeit ausgelegt sind.

Nach Erkenntnissen des Verfassers wurde der nächste Schritt innerhalb der computergestützten FMEA-Verwendung von der Baumstruktur zur Erstellung einer Ontologie bisher nicht vollzogen. Somit wurde bisher dieser Schritt auch noch nicht unter der wissenschaftlichen Basisanforderung der Nachvollziehbarkeit untersucht.

Es fehlte bisher noch ein systematisches Verfahren, um das bereits umfangreich verfügbare, aber in hierarchischen Strukturen „gekapselte" FMEA-Wissen in die (Wissensrepräsentations)-Form von Ontologien zu transformieren. Im vorangehenden Kapitel wurde ein solches systematisches Vorgehen in Form des OntoFMEA-Vorgehensmodells vorgestellt. Um die praktische Anwendbarkeit dieses systematischen Transformationsverfahrens zu demonstrieren, wird anhand der beispielhaften Anwendungsfälle der Karl Schumacher Maschinenbau GmbH das Ergebnis der Verfahrensanwendung als FMEA-Ontologie im vorliegenden Kapitel vorgestellt.[805]

Die hier vorgestellte FMEA-Ontologie geht über den Stand der Forschung deutlich hinaus.[806] Die FMEA-Ontologie ist eine Domänen-Ontologie, die für einen Teilbereich des Qualitätsmanagements – die Fehleranalyse (insbesondere die Fehlermöglichkeits- und Einflussanalyse) – Gültigkeit besitzt.[807]

Die Gesamtdarstellung der FMEA-Ontologie im erweiterten Sinn mit dem Ausschnitt einer Wissensbasis (den zugehörigen Instanzen) als zusammenhängendes Artefakt findet sich im Anhang A.2, S. 336 ff.

805) Zu den hier beispielhaft verwendeten Anwendungsfällen siehe Kapitel 3.4, S. 70 ff.

806) Zu den Wirkungen des hier vorgestellten Ansatzes siehe Kapitel 9.1.3, S. 281 ff. Ein direkter Vergleich mit dem Ansatz von Lee wird ausgeschlossen, weil zum einen das Ontologieverständnis von Lee nicht dem hier vertretenen Anspruch gerecht wird und weil zum anderen Lee bei seinen veröffentlichten Ergebnissen vage bspw. hinsichtlich Umfang und Komplexität seiner Ontologie bleibt. Zusätzlich wird die Berücksichtigung eines systematischen Vorgehens bei der Ontologieentwicklung innerhalb seiner Ausführungen nicht deutlich. Siehe hierzu auch Kapitel 2.2, S. 34 f.

807) Zu Arten von Ontologien siehe Kapitel 4.1.2, S. 76 f.

7.2 Meta-Meta-Ebene

7.2.1 Beschreibung der Ebene

Auf der Meta-Meta-Ebene werden die Grundlagen für die Ontologie festgelegt. Insbesondere die Festlegung auf eine Repräsentationssprache mit ihren spezifischen Modellierungsprimitiven zur Darstellung von Ontologien und die durch die Definition von Ontologien vorgegebenen Modellierungsprimitive schlagen sich in ihrem Einfluss auf die Meta-Meta-Ebene nieder. In der Hauptsache lassen sich hier die einflussreichen Modellierungsprimitive hinsichtlich des verwendeten Alphabets, der betrachteten Objekte und ihrer Darstellungsweise unterscheiden.[808]

7.2.2 Modellierungsprimitive von F-Logic

Wie bereits in Kapitel 1.1, Seite 95, dargestellt, existieren unterschiedliche Sprachen zur Repräsentation von Ontologien. Die Sprache F-Logic, die in dieser Arbeit Verwendung findet, kann als Resultat einer Kombination von Elementen der konzeptuellen Modellierung mit Elementen logischer Kalküle angesehen werden. F-Logic stellt eine Integration frame-basierter Sprachen[809] mit der Prädikatenlogik erster Ordnung dar. F-Logic umfasst die strukturellen Aspekte objektorientierter und frame-basierter Sprachen, wie z. B. Objektidentität, Vererbung, polymorphe Typen, Kapselung und Abfragemechanismen.[810] Weitere Vorteile liegen in der einfachen Lesbarkeit der formalen Ontologie, der prägnanten Darstellung der formalen Ontologie und der einfacheren und intuitiveren Erstellbarkeit der Formalisierung in F-Logic gegenüber anderen Sprachen.[811]

---

808) Die in diesem Unterkapitel gemachten Angaben dienen lediglich dem Fokus der Untersuchungen, d. h. sie erheben keineswegs einen Anspruch auf Vollständigkeit. Insbesondere zu Ausführungen über Typen, Signaturen u. ä. sei auf die einschlägigen Quellen verwiesen.

809) Für weitergehende Informationen zum Frame-Konzept vgl. Minsky (1975). Obwohl der Begriff *F-Logic* an das Frame-Konzept aus der KI angelehnt wurde, stammen Terminologie und wesentliche Konstrukte von F-Logic eher aus der objektorientierten Programmierung. Der objektorientierte Ansatz in der Programmierung und das Frame-Konzept besitzen weitgehende gemeinsame Merkmale.

810) Vgl. Kifer, Lausen et al. (1995), S. 741.

811) Vgl. Friedland, Allen (2003), S. 14. Die Autoren beziehen sich dabei in ihren allgemein gehaltenen Aussagen über Repräsentationssprachen insbesondere auf die Repräsentationssprachen F-Logic (vgl. Kifer, Lausen et al. (1995); Angele, Lausen (2004)), KM (vgl. Clark, Porter (1999)) und CycL (siehe Kapitel 4.2.2.1.4, S. 106).

Das hier verwendete Alphabet für die Darstellung einer Ontologie in F-Logic besteht aus einer Menge von:[812]

- Objektkonstruktoren (F),
- Variablen (V),
- Hilfssymbolen (->, ->>, =>, =>>, (,), [,], ", : und .) sowie
- Operatoren aus der Prädikatenlogik erster Stufe[813] ($\neg$, $\wedge$, $\vee$, $\rightarrow$, $\leftrightarrow$, $\forall$ und $\exists$).

Die Basis-Konstrukte der Sprache F-Logic sind Objekte, die als Terme aus der Menge F der Objektkonstruktoren zusammengesetzt und als Funktionssymbole aufgefasst werden.[814] Dabei repräsentieren Objekte Entitäten der „realen" Welt. Objekte können zu Klassen zusammengefasst und über Methoden miteinander in Relation gesetzt werden.[815] Konstanten werden als 0-stellige Funktionssymbole dargestellt. Hervorzuheben dabei ist, dass Konstanten, Klassen und Methoden reifiziert und so selbst als Objekte aufgefasst werden.[816] Terme über $F \cup V$ kennzeichnen Objekte, Methoden oder Klassen und werden ID-Terme genannt.[817] Sie werden rechnerintern, d. h. für den Benutzer einer Computeranwendung unsichtbar[818], durch einen

812) Vgl. hierzu auch Kifer, Lausen et al. (1995), S. 12; Ludäscher, Himmeröder et al. (1998); Erdmann (2001), S. 89 ff. In dieser Arbeit wird auf einige Bestandteile des ursprünglichen Alphabets gemäß Kifer, Lausen und Wu von vorneherein verzichtet, weil sie zum einen nicht für die Modellierung der FMEA-Ontologie notwendig sind (wie bspw. die Parameterfunktion „@"; vgl. Angele, Lausen (2004), S. 33) und weil sie zum anderen von der Inferenzmaschine OntoBroker nicht unterstützt werden (wie bspw. die Möglichkeit, vererbliche und nicht vererbliche Instanzenrelationen zu modellieren; vgl. May (2000), S. 9).
Ein Alphabet wird als endliche Menge von Symbolen als Bestandteil einer formalen Sprache angesehen, die zu Ausdrücken (Termen) zusammengesetzt werden können. Objektkonstruktoren, Variablen und Operatoren werden als Unterfälle von Symbolen deklariert.

813) Siehe hierzu auch die Ausführungen in Unterkapitel 4.2.1.2.2, S. 99 ff. In F-Logic werden einige Symbole abweichend zur üblichen Darstellung innerhalb der Prädikatenlogik dargestellt. Bspw. werden die Symbole $\forall$ und $\exists$ durch die Symbole FORALL bzw. EXIST dargestellt. Des Weiteren wird der Implikationspfeil in der umgekehrten Richtung mit gleicher Bedeutung verwendet, d. h. Regeln in F-Logic werden von rechts nach links bearbeitet.

814) Vgl. Ludäscher, Himmeröder et al. (1998), S. 3; Angele, Lausen (2004), S. 32, und Ontoprise (2004), S. 4. Funktionssymbole sind bspw. Objektbezeichner, Methoden- und Klassennamen (vgl. Erdmann (2001), S. 90).

815) In Kifer, Lausen et al. (1995) und auch Ontoprise (2004) sowie weiteren Quellen werden für die hier verwendeten Begriffe Konzepte und Relationen synonym die Begriffe Klassen bzw. Methoden verwendet. Grund hierfür ist, dass diese Quellen einen starken Bezug zum Paradigma der objektorientierten Programmierung aufweisen, in dem diese Begriffe vorwiegend Verwendung finden. Im Gegensatz dazu wird in dieser Arbeit einem „linguistischen" Paradigma gefolgt, das Begriffe in Konzepte und sie verbindende Relationen unterscheidet; siehe hierzu auch die Erläuterungen zu Ontologien in Unterkapitel 1, S. 74 ff. Lediglich bei der Programmierung der Java Server Pages für die prototypische Softwareanwendung wird in dieser Arbeit auf das „informationstechnische" Paradigma zurückgegriffen.
In F-Logic wird die aus der objektorientierten Programmierung bekannte Unterscheidung von Attributen und Methoden nicht gemacht. Stattdessen wird vereinheitlichend nur von Methoden ausgegangen (vgl. Erdmann (2001), S. 90).

816) Vgl. Ontoprise (2004), S. 3.

817) Vgl. Erdmann (2001), S. 90.

818) Vgl. May (2000), S. 6.

Objektidentifizierer (den Namen) eindeutig, unabhängig von ihren möglichen Eigenschaften, repräsentiert.[819] Üblicherweise werden Namen für Objekte mit dem ersten Buchstaben klein geschrieben (und dabei mit dem Zeichen „_" durchgekoppelt), in Hochkommata (als String) eingeschlossen oder als ganze Zahlen dargestellt (als Integer-Wert). Variablen werden mit dem ersten Buchstaben groß geschrieben.

Zusammenfassend werden folgenden Modellierungsprimitive der Objektorientierung bei der Verwendung von F-Logic als gegeben angenommen:

- *Objekte*, die Entitäten der Domäne repräsentieren und synonym als Instanzen bezeichnet werden, sofern ihre Zugehörigkeit zu einer Klasse/einem Konzept ausgedrückt werden soll;

- *Klassen* als Menge von Objekten, die im Kontext von Ontologien synonym als Konzepte bezeichnet werden;

- *Methoden* als Mengen von zweistelligen Beziehungen zwischen Objekten oder zwischen Klassen, die im Kontext von Ontologien synonym als Relationen bezeichnet werden.

Als deduktive objektorientierte Datenbanksprache soll F-Logic für die Darstellung von Ontologien verwendet werden. Für den hier angestrebten Zweck der Ontologieerstellung werden im Folgenden zunächst die Modellierungsprimitive Objekte, Klassen und Methoden von F-Logic eingehend zu Instanzen, Konzepten und Relationen transformiert, die wiederum als Modellierungsprimitive einer Ontologie und ihrer Wissensbasis anzusehen sind.[820]

7.2.3 Repräsentation von Ontologien in F-Logic

7.2.3.1 Daten-F-Atome

In F-Logic wird die Anwendung einer Methode auf Objekte als *Daten-F-Atom* ausgedrückt.[821] Ein (Start-)Objekt wird dabei mit einem (Ziel-)Objekt über eine Methode verbunden. Hinsichtlich der Kardinalität der Methode wird dabei lediglich zwischen einwertigen (auch funktionalen) und mehrwertigen (auch mengenwertigen) Ausprägungen von Methoden unterschieden. Bei einer einwertigen Ausprägung wird festgelegt, dass ein (Start-)Objekt mit genau einem weiteren (Ziel-)Objekt verbunden wird. Bei einer mehrwertigen Ausprägung wird ein

819) Dieser Sachverhalt stellt ein sehr anschauliches Beispiel für einen Bestandteil eines rechnerinternen Modells dar, das mit der Abbildung 41, S. 195, eingeführt und erläutert wurde.

820) Siehe hierzu auch Kapitel 2.1.2.4 und Kapitel 4. Streng genommen ergibt sich aus diesen Ausführungen, dass die betrachtete Meta-Meta-Ebene mindestens noch eine weitere Ebenenunterscheidung enthalten würde, nämlich zwischen den Modellierungsprimitiven Objekt, Klasse, Methode (Semantik von F-Logic) und den Modellierungsprimitiven Instanz, Konzept, Relation der Ontologie im weiteren Sinn. Der Verfasser glaubt jedoch, der Forderung nach Systematik Genüge getan zu haben. Um den Leser nicht weiter zu überfrachten mit Ebenen, die wiederum auf Ebenen rekurrieren, werden alle denkmöglichen weiteren „höheren" Ebenen zur Meta-Meta-Ebene zusammengefasst.

821) Vgl. Angele, Lausen (2004), S. 33.

(Start-)Objekt über eine Methode mit mindestens einem (Ziel-)Objekt verbunden, die mit einer geschweiften Klammer zusammengefasst werden.

Die Objekte entsprechen in der ontologiebasierten Wissensrepräsentation den Instanzen (`i`), die einer Ontologie als Wissensbasis zugeordnet werden.[822] Eine Methode gemäß F-Logic entspricht einer Relation und wird mittels der Hilfssymbole `->` (einwertig) und `->>` (mengenwertig) gekennzeichnet. Es sind sowohl Relationen auf Instanzen-Ebene als auch auf Konzeptebene darstellbar.

Der erste Ausdruck in F-Logic-Kodierung im folgenden Kasten bezieht sich auf eine einwertige Relationsausprägung (Beziehung) und die zweite Aussage auf eine mehrwertige Relationsausprägung (Beziehung) hinsichtlich der Instanz i_0. Einzelne Ausdrücke in F-Logic werden mittels des Hilfssymbols `.` (einfacher Punkt) voneinander getrennt:

$i_0[rel_a\text{->}i_1].$

$i_0[rel_b\text{->>}\{i_1,i_2,\ldots,i_n\}].$

Die Platzhalter `i` und `rel` stehen jeweils für ID-Terme. Für den ersten Fall bedeutet das Daten-F-Atom, dass einer Instanz i_0 über eine Relation rel_a genau eine Instanz i_1 zugeordnet wurde. Im zweiten Beispiel des Kastens führt eine mehrwertige Relation rel_b zu verschiedenen resultierenden Instanzen i_1, i_2,..., i_n, die mit einer geschweiften Klammer zu einer Menge zusammengefasst werden.

7.2.3.2 Ist_ein-F-Atome

In F-Logic kann ein Objekt einer Klasse zugeordnet werden. Dabei ist es auch möglich, ein Objekt mehreren Klassen zuzuordnen. Ein zugeordnetes Objekt wird als Instanz der Klasse bezeichnet.

Entsprechend gilt für die Darstellung einer Ontologie und ihrer zugeordneten Wissensbasis, dass Instanzen Konzepten zugeordnet werden können. Die Zuordnung von Instanzen zu Konzepten erfolgt in F-Logic anhand von *Ist_ein-F-Atomen*, die mit dem Hilfssymbol `:` (einfacher Doppelpunkt) ausgedrückt werden.

Die Instanz *i* eines Konzepts *c* wird somit folgendermaßen definiert:

```
i:c.
```

822) Genau genommen besteht eine Wissensbasis nicht aus „Haufen" von Instanzen, sondern aus Formeln (oder Frames u. ä.), die vollständig instanziiert sind. Für alle Konzepte und Relationen sind deren aktuell gültigen Extensionen als Instanzenmenge spezifiziert. Alle „freien" (d. h. nicht durch Quantoren gebundenen) Variablen sind durch Instanzen substituiert und jede Formel ist wahrheitsdefinit.

7.2.3.3 Subklassen-F-Atome

Subklassen-F-Atome drücken die Subklassenbeziehungen zweier Klassen in F-Logic aus, d. h. eine Klasse ist eine Unterklasse einer „höher stehenden“ Klasse.[823] Da Klassen ebenfalls Objekte in F-Logic darstellen, lässt sich feststellen, dass über eine Methode (bspw.: subklasse; die selbst wiederum nach Reifizierung ein Objekt darstellt) ein (Ziel-)Objekt einem (Start-) Objekt als Unterklasse zugeordnet wird. In F-Logic wird zur Darstellung das Hilfssymbol `::` (doppelter Doppelpunkt) eingesetzt.

Die Subklassenbeziehungen der Objektorientierung entsprechen den Abstraktionsrelationen einer Ontologie. Für die Menge der Abstraktionsrelationen innerhalb der hierarchischen terminologischen Relationen R^H zwischen Konzepten[824] bietet es sich an, diese gesondert darzustellen, um ihre besondere Relevanz zu berücksichtigen: da die Abstraktionsrelationen in ihren Eigenschaften den Ist_ein-F-Atomen entsprechen, indem sie Merkmale von Instanzen auf höheren Ebenen generalisieren und in einer Hierarchie darstellen, werden Unterkonzeptbeziehungen ähnlich dargestellt. Es ergibt sich die folgende Notation:

```
c1::c2.
```

Das Konzept c_1 gilt als Unterkonzept von c_2 und erbt damit alle Relationen von c_2.[825]

7.2.3.4 Signatur-F-Atome

Signatur-F-Atome in F-Logic legen fest, welche Methoden auf die Instanzen einer bestimmten Klasse anwendbar sind.[826] Dabei wird festgelegt, von welchem Typ die Objekte sein werden, die unter Anwendung der Methode auf die Instanzen einer Klasse entstehen.[827] Es wird wiederum zwischen einwertigen und mengenwertigen Methoden unterschieden.

Mit Hilfe der Signatur-F-Atome wird in der Ontologie die Menge der nicht-hierarchischen terminologischen Relationen R^R sowie die Menge der Bestandsrelationen aus R^H festgelegt.[828] Für die Darstellung der Menge der nicht-hierarchischen terminologischen Relationen und der Bestandsrelationen einer Ontologie ergibt sich hieraus unter Verwendung der Hilfssymbole => (einwertig) und =>> (mehrwertig) die folgende Kodierung:

---

823) Zu einer Erläuterung der Prinzipien für die hierarchische Darstellung von Klassen/Konzepten siehe die Ausführungen zum Menelas-Ansatz in Kapitel 5.1.2.3.2.1, S. 149 f.

824) Zu Erläuterung der Menge R^H siehe Kapitel 4.3, S. 114 f. Siehe ebenda zur besonderen Relevanz dieser Relationen.

825) Relationen wurden bislang nur auf Instanzen angewendet. In diesem Sinne bedeutet die Aussage dieses Satzes einen Vorgriff auf das nächste Unterkapitel, indem gezeigt wird, dass Relationen „analog“ allgemein auch auf Konzepte eingeführt werden.

826) Vgl. May (2000), S. 9.

827) Vgl. Erdmann (2001), S. 91.

828) Siehe hierzu Kapitel 4.3, S. 114 f.

```
c₁[rel_a=>c₂].
c₁[rel_b=>>c₂].
```

Für den ersten Fall bedeutet das Signatur-F-Atom, dass einer Instanz des Konzepts c_1 über eine Relation rel_a genau eine Instanz des Konzepts c_2 zugeordnet wird. Im zweiten Beispiel des Kastens verbindet die Relation rel_b eine Instanz des Konzepts c_1 mit mindestens einer Instanz des Konzepts c_2.

Des Weiteren werden mittels der Signatur-F-Atome einige a priori feststehende Modellierungsprimitive für die Ontologie verankert. Es handelt sich bei den Modellierungsprimitiven um Datentypen für Variablen oder Instanzen.[829] In der FMEA-Ontologie wird zwischen den drei Datentypen BOOLEAN, STRING oder INTEGER unterschieden.[830] Ein BOOLEsches Objekt kann nur eine der zwei möglichen Konstanten *true* oder *false* annehmen. Ein Objekt vom Typ STRING kennzeichnet eine Kette von beliebigen Zeichen, die in Hochkommata eingeschlossen wird. Ein INTEGER-Objekt kennzeichnet eine ganze Zahl. Es wird festgelegt, dass die Instanzen („Daten"), die über eine Relation den Instanzen eines Konzepts zugeordnet werden, Instanzen der Datentypen BOOLEAN, STRING, INTEGER sein müssen. Folgende Kodierung ergibt sich beispielhaft für die drei genannten Datentypen:

```
c₁[rel_a=>BOOLEAN].
c₁[rel_b=>STRING].
c₁[rel_c=>INTEGER].
```

7.2.3.5 F-Formeln

7.2.3.5.1 Molekulare Formeln

Die in den vorherigen Ausführungen so genannten *F-Atome* umfassen Entitäten, vergleichbar mit Objekten, Klassen und Methoden aus der objektorientierten Programmierung. F-Atome sind die kleinsten wahrheitsdefiniten Ausdrücke, die sich mit den Ausdrucksmitteln von F-Logic formulieren lassen. Ein Ausdruck ist im Rahmen einer vorausgesetzten Logik wahrheitsdefinit, wenn ihm „Kraft seiner Form" ein eindeutiger Wahrheitswert (hier: wahr/nicht wahr) zugeordnet werden kann. Um zu gehaltvolleren Aussagen zu gelangen, werden die einzelnen F-Atome zu F-Formeln zusammengefügt.

829) Beliebige Bitmuster können von einem Computerspeicher aufgenommen werden. Die Bedeutung einer Folge von Bits hängt von ihrer Verwendung ab. Ein Datentyp ist ein Schema für die Verwendung einer Folge von Bits, um bestimmte Werte darzustellen. Dabei kommen nicht nur Zahlen, sondern auch weitere Schriftzeichen als mögliche Werte in Betracht.

830) Diese Datentypen werden auch als „flache" oder „primitive" Datentypen bezeichnet. In der objektorientierten Programmierung werden hierunter Datentypen subsumiert, die *ab initio* existieren, d. h. die nicht aus anderen Datentypen zusammengesetzt werden. Sie werden oftmals auch als Attribute bezeichnet.

Die simpelsten Formeln in F-Logic werden molekulare F-Formeln genannt (F-Moleküle).[831] Sie werden vornehmlich genutzt, um die Wissensdarstellung verlustfrei zu verdichten. Dabei werden mehrere F-Atome zu einem F-Molekül angeordnet (Listen von Relationen werden dabei durch Semikola getrennt). Es handelt sich dabei um die konjunktive Verknüpfung von F-Atomen.[832] So lässt sich das folgende F-Molekül in die darauf folgenden F-Atome spalten.

```
c_5960_a_fmea_bedienpult:fmea[wurde_erstellt->_03062000;
                        untersucht_systemelement->c_5960_a_bedienpult].
```

Zugehörige F-Atome:

```
c_5960_a_fmea_bedienpult:fmea.
c_5960_a_fmea_bedienpult[wurde_erstellt->_03062000].
c_5960_a_fmea_bedienpult[untersucht_systemelement->c_5960_a_bedienpult].
```

F-Moleküle werden von links nach rechts betrachtet. Durch geeignete Klammersetzung lässt sich das gewünschte zu repräsentierende Wissen darstellen. Würde man im dargestellten Beispiel runde Klammern vor das Konzept `fmea` und hinter das Ende des Ausdrucks setzen, änderte sich der Inhalt des repräsentierten Wissens. Mit den Klammern wird ausgesagt, dass das Konzept `fmea` Relationen zu den Instanzen `_03062000` und `c_5960_a_ bedienpult` besitzt. Mit den Klammern beinhaltet das Molekül das Wissen, dass lediglich die Instanz `c_5960_a_fmea_bedienpult` die beiden Relationen besitzt.[833] Es ist möglich, sehr komplexe Informationseinheiten zu generieren, die dementsprechend zeitintensiv durch Dritte nachzuvollziehen sind.

Die gemachten Ausführungen gelten analog auch für die Signatur-F-Atome (mit den Hilfssymbolen => und =>>), d. h. auf der Ebene von Konzepten und Relationen.[834]

7.2.3.5.2 Komplexe Formeln

Unter Zuhilfenahme einiger Operatoren der Prädikatenlogik (Junktoren und Quantoren) werden F-Moleküle zu komplexen F-Formeln zusammengefügt. Dabei gilt:[835]

- Molekulare Formeln sind F-Formeln (s. o.);
- $\varphi \vee \psi$, $\varphi \wedge \psi$ und $\neg\varphi$ sind F-Formeln, wenn φ und ψ F-Formeln sind;

831) Vgl. Kifer, Lausen et al. (1995), S. 754. Zusätzlich werden noch Ist_ein-F-Atome und Subklassen-F-Atome von den Autoren als simpelste Formeln genannt, weil diese streng genommen schon durch ihre Darstellungsform eine Verdichtung bedeuten.

832) Vgl. Erdmann (2001), S. 91.

833) Siehe hierzu auch Angele, Lausen (2004), S. 37.

834) In dem nachfolgenden Unterkapitel „Meta-Ebene“ wird auf molekulare Formeln noch weitergehend eingegangen, so dass an dieser Stelle lediglich hierauf verwiesen sei (S. 232 ff.).

835) Vgl. Kifer, Lausen et al. (1995), S. 14.

- ∀ X: φ(X), ∃ Y: ψ(Y) sind F-Formeln, wenn φ, ψ F-Formeln und X, Y Variablen sind.

Diese komplexen Formeln werden genutzt, um die Regeln einer Ontologie zu implementieren. Dabei gilt, dass eine logische Aussage, die mit einem Subjugatspfeil (<-) modelliert wird, als Inferenzregel der Ontologie bezeichnet wird.[836] Rechts vom Pfeil steht der Regelkörper (*body*) mit den Prämissen einer Schlussfolgerung und links vom Implikationspfeil findet sich der Regelkopf (*head*) mit der Konklusion einer Schlussfolgerung. Eine Inferenzregel in F-Logic wird somit von rechts nach links „gelesen". Nachfolgend sei ein Beispiel gegeben:[837]

```
FORALL X,Y X[is_direct_sub->>Y] <- X::Y AND NOT EXISTS Z(Z::Y AND X::Z).
```

Die Regel besagt, dass, wenn es ein Konzept `X` gibt, dieses ein Unterkonzept von `Y` ist und gleichzeitig kein Konzept `Z` existiert, für das gilt, dass es Unterkonzept von `Y` und Oberkonzept von `X` ist, dann werden die Konzepte `X` und `Y` über die Relation `is_direct_sub` miteinander verbunden, d. h., dass X direktes Unterkonzept von Y ist.[838]

Des Weiteren ist es möglich, Symmetrieregeln innerhalb der Ontologie festzulegen. Eine Symmetrieregel legt fest, welche Relationen als symmetrisch angesehen werden.[839]

---

836) Zu beachten ist hierbei, dass in der klassischen Logik das Subjugat $\varphi \leftarrow \psi$ lediglich die Kurzform der logischen Aussage $\varphi \vee \neg\psi$ darstellt. Aus Gründen der einfacheren Lesbarkeit wird im folgenden Text mit dem Subjugat gearbeitet.

837) Wie bereits in Fn. 813, S. 222, erwähnt wurde, erfolgt eine Transformation der prädikatenlogischen Symbole bei der Repräsentation von Wissen in F-Logic. In der nachfolgenden Regel werden für die Symbole ∀, ∧, ¬ und ∃ die Entsprechungen FORALL, AND, NOT bzw. EXISTS verwendet. Die vorliegende Regel kann in einer Ontologie bei der Verwendung der Inferenzmaschine OntoBroker weggelassen werden, weil diese mit einem Built-In (`directsub_`) bereits die gewünschte Funktionalität bereithält. Ein Built-In entspricht einem „fest eingebauten" Modellierungsprimitiv seitens OntoBroker, das bei der Darstellung einer Ontologie zusätzlich zu den Modellierungsprimitiven, die durch die Wissensrepräsentationssprache vorgegeben werden, berücksichtigt werden kann. D. h. es wird von der Inferenzmaschine verarbeitet. Zweck der Built-Ins ist es bspw., den Einsatz aufwändiger Infernzregeln zu vereinfachen, indem einzelne Konstrukte abkürzend zur Verfügung gestellt werden.

838) Bei der Regel handelt es sich um ein besonders „trickreiches" Exemplar. Mit Hilfe dieser Regel soll die Konzepthierarchie (Abstraktionsrelationen) einer Ontologie unter Verwendung der Inferenzmaschine abgefragt werden. Hierzu ist es notwendig, Relationen zwischen Konzepten zu beschreiben. Es soll nicht ausgedrückt werden, dass Relationen zwischen Instanzen zweier Konzepte bestehen können, wie es bei einer Formulierung mittels X[is_direct_sub=>>Y] der Fall sein würde. Ermöglicht wird dieser „Trick" durch eine Vorgabe der Wissensrepräsentationssprache F-Logic, die bei Objekten bspw. keinen Unterschied zwischen Instanzen und Konzepten macht, d. h. ein Objekt kann in einem Fall ein Konzept darstellen und gleichzeitig in einem anderen Fall eine Instanz darstellen (Reifizierung). Wie bereits in der voranstehenden Fn. erläutert wurde, existiert in der hier verwendeten Inferenzmaschine OntoBroker ein Built-In, das den Zweck der Inferenzregel ebenfalls zu erfüllen hilft. Weil bei der Anwendung eines solchen Built-Ins zu erwarten ist, dass dieses performanter angewendet werden kann als die explizite Regel, wird im Folgenden auf die bereitgestellten Built-Ins zurückgegriffen. Die bei dieser Regel nicht ganz saubere Trennung von Ontologie und Wissensbasis soll neben der Erläuterung eines Regelaufbaus in F-Logic der Verdeutlichung der Möglichkeiten zur Formulierung von Wissen in F-Logic dienen.

839) In dem nachfolgenden Unterkapitel „Meta-Ebene" wird noch weitergehend auf diese Art der Regeln eingegangen, so dass an dieser Stelle lediglich hierauf verwiesen sei (S. 232 ff.). Eine Symmetrieregel wird als „doppeltes" Subjugat, d. h. als Schlussfolgerung, die in beiden Richtungen gültig ist, angesehen. Deshalb kann der Begriff „Regel" weiter verwendet werden.

Der Verfasser konnte für den Fokus dieser Arbeit keine relevanten Integritätsregeln feststellen, deshalb wurde für die Erstellung der FMEA-Ontologie auf die Berücksichtigung von Integritätsregeln verzichtet. Aus diesem Grund erübrigt sich die Vorstellung solcher Regeln an dieser Stelle.

Mit den vorgestellten Modellierungsprimitiven der Meta-Meta-Ebene wird es möglich, die Meta-Ebene mit den Modellierungsprimitiven einer FMEA (gemäß Formblatt) zu entwickeln. Um eine prototypische Anwendung (OntoFMEA) realisieren zu können, muss zunächst noch die Möglichkeit der Abfrage, d. h. eines Wissenszugriffs auf die Ontologie, realisiert werden.

7.2.4 Abfragen in F-Logic

Eine Abfrage (*query*) in F-Logic entspricht einer Inferenzregel ohne Kopf. Die Antwort einer Abfrage ergibt die Konklusion. Damit entspricht eine Abfrage der Angabe von Prämissen. Ein kurzes Beispiel verdeutlicht diesen Zusammenhang:

```
FORALL X <- X:systemelement.
```

Die dargestellte beispielhafte Abfrage sucht nach den Instanzen in der Wissensbasis, die dem Konzept `systemelement` zugeordnet worden sind. Deutlich wird, dass der Kopf fehlen würde, wenn man annehmen würde, dass es sich bei dem Beispiel um eine Regel handeln soll.

Die Bearbeitung der Abfragen in OntoFMEA erfolgt bei der hier vorgestellten Version im Hintergrund ohne Einsichtsmöglichkeit seitens des Benutzers. Deshalb wird an dieser Stelle nicht intensiver auf die Abfragemöglichkeiten eingegangen, jedoch gilt, dass grundsätzlich dieselben komplexen Formeln für die Abfragen benutzt werden können wie bei den Regeln.

7.3 Meta-Ebene

7.3.1 Beschreibung der Ebene

Auf der Meta-Ebene werden die wichtigsten Konzepte und Relationen der FMEA-Ontologie festgelegt.[840] Um eine vollständige FMEA-Ontologie zu entwickeln, müssen alle konstituierenden Konzepte und Relationen aus FMEA-Formblättern Eingang in die Entwicklung finden. Diese Hauptkonzepte entsprechen den Modellierungsprimitiven, die durch das Instrument FMEA vorgegeben werden. Diese Modellierungsprimitive ermöglichen die anschließende FMEA-Erstellung auf Basis der FMEA-Ontologie. Die ermittelten Konzepte werden zunächst natürlichsprachlich beschrieben und anschließend in F-Logic repräsentiert.[841] An dieser Stelle fließen die Erkenntnisse aus Kapitel 3, S. 35 ff., ein. Insbesondere die Ausfüh-

---

840) Diese wichtigsten Konzepte werden im weiteren Verlauf der Arbeit auch „Hauptkonzepte" genannt. Vereinfacht dargestellt, handelt es sich bei der Menge der Hauptkonzepte um die Gesamtheit aller Konzepte, die als konstituierend für die Durchführung einer FMEA angesehen werden und die damit zur Abbildung des Wissens einer FMEA mindestens notwendig sind.

841) Dieses Vorgehen entspricht bspw. dem Ansatz von Uschold und King, siehe hierzu auch den Enterprise-Model-Ansatz, Kapitel 5.1.2.3.1.2, S. 137 ff.

rungen zum Phasenmodell der FMEA-Durchführung, S. 47, und die Vorgaben für ein Formblatt zur Durchführung der System-FMEA, S. 60, finden Berücksichtigung bei der Identifikation der wichtigsten Konzepte und Relationen einer FMEA-Ontologie.

7.3.2 Hauptkonzepte der Meta-Ebene der FMEA-Ontologie

Die FMEA-Ontologie wird mittels eines Top-Down-Ansatzes, d. h. beginnend mit den „obersten" Konzepten, entwickelt. Aufgrund der Modellierungsprimitive der Meta-Meta-Ebene, die durch die eingesetzte Inferenzmaschine vorgegeben werden, ist es notwendig, ein einzelnes oberstes Konzept mit der Bezeichnung DEFAULT_ROOT_-CONCEPT als gemeinsamen Startpunkt festzulegen.[842]

Zur obersten Konzeptebene der Meta-Ebene der FMEA-Ontologie gehören die Konzepte `fmea`, `systemelement`, `funktion`, `fehler`, `fehlertupel`, `massnahme`, `datum` und `verantwortungstraeger`. Diese Konzepte werden auch als Hauptkonzepte bezeichnet; sie sind mindestens notwendig, um das Wissen einer FMEA repräsentieren zu können.

Die Abbildung 52 verdeutlicht die Zusammenhänge auf der obersten Konzeptebene, indem für die genannten Konzepte zusätzlich die wichtigsten Relationen der Konzepte zueinander aufgezeigt werden.[843]

Die Relationen, die mit den einzelnen Konzepten verbunden sind, werden im Folgenden näher beschrieben. Als Grundlage für die Ausführung der Relationen und Konzepte werden insbesondere die Ausführungen in Kapitel 3.1.4, S. 46 ff., berücksichtigt.

842) Um die Besonderheit dieses Konzepts (DEFAULT_ROOT_CONCEPT) hervorzuheben, wird es als einzigstes Objekt mit großen Buchstaben geschrieben. Der Vereinfachung halber werden dieses einzelne oberste Konzept und die direkt mit diesem Konzept als Unterkonzepte Verbundenen zur obersten Konzeptebene zusammengefasst. Alternativ ließe sich das DEFAULT_ROOT_CONCEPT auch auf einer gedachten „nullten" Ebene ansiedeln.

843) Bei der vorliegenden Darstellung der Hauptkonzepte wird auf die nachfolgenden Ausführungen vorgegriffen. So wäre es im herkömmlichen Verständnis zur FMEA-Anwendung unmittelbar wünschenswert, bspw. Konzepte zu „Fehlerursache" und „Fehlerfolge" festzulegen. Aufgrund später erläuterten Überlegungen kann und wird jedoch auf diese beiden Konzepte verzichtet. Damit enthält die Abbildung lediglich die Konzepte, die tatsächlich einer Verwendung zugeführt wurden. Siehe zu den Ausführungen zu „Fehlerursache" und „Fehlerfolge" Kapitel 7.3.3.4, S. 236 f.

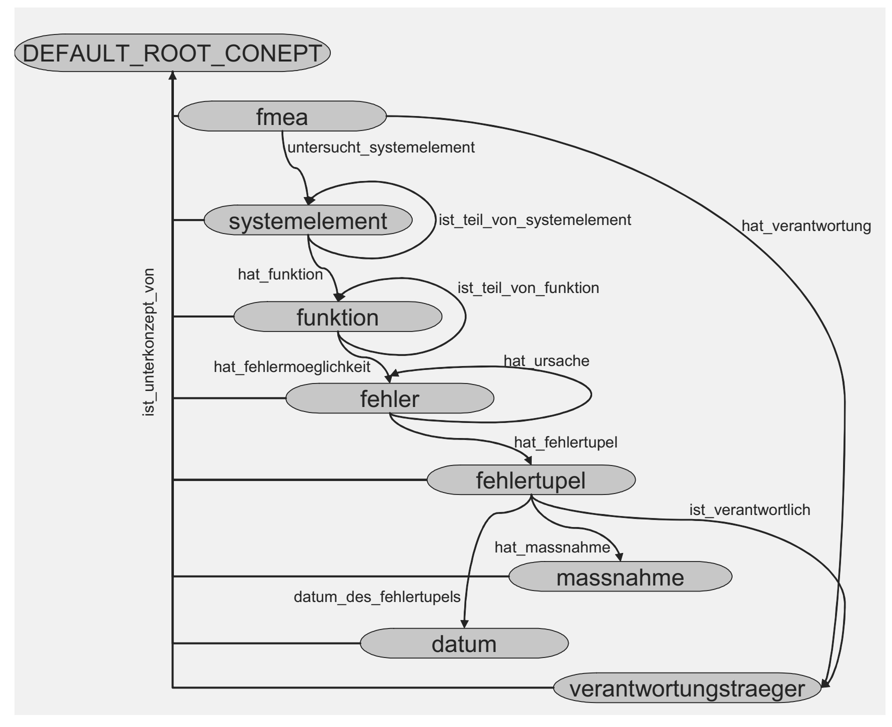


Abbildung 52: Oberste Konzeptebene der FMEA-Ontologie

In der F-Logic-Kodierung ergibt sich folgendes Bild:[844]

```
fmea::DEFAULT_ROOT_CONCEPT.
systemelement::DEFAULT_ROOT_CONCEPT.
funktion::DEFAULT_ROOT_CONCEPT.
fehler::DEFAULT_ROOT_CONCEPT.
fehlertupel::DEFAULT_ROOT_CONCEPT.
massnahme::DEFAULT_ROOT_CONCEPT.
datum::DEFAULT_ROOT_CONCEPT.
verantwortungstraeger::DEFAULT_ROOT_CONCEPT.
      unternehmen::verantwortungstraeger.
      person::verantwortungstraeger.
```

844) Für das Konzept verantwortungstraeger werden an dieser Stelle bereits zwei Unterkonzepte (unternehmen und person) festgelegt. Streng genommen werden diese zur zweiten Konzeptebene gezählt, weil sie bereits Unterkonzepte darstellen. Aufgrund der geringen Komplexität der Kodierung bietet es sich an, die Konzepte bereits hier festzulegen, um dem in F-Logic weniger bewanderten Leser noch weiter in die Syntax der Sprache einzuführen.

7.3.3 Definitionen der Hauptkonzepte der Meta-Ebene der FMEA-Ontologie

Zu den einzelnen Hauptkonzepten werden in den folgenden Unterkapiteln die zugehörigen F-Moleküle festgelegt, die die Hauptkonzepte definieren. Die Definition eines Konzepts erfolgt „intensional“ durch die Darstellung der Menge aller Relationen („Merkmale“), in denen das Konzept zu anderen Konzepten steht. Ein Hauptkonzept wird in der FMEA-Ontologie durch genau ein F-Molekül definiert, das sich nur aus Signatur-F-Atomen zusammensetzt, in denen das Hauptkonzept selbst jeweils an erster Stelle der enthaltenen Relationen steht und dabei vollständig alle Signatur-F-Atome, in denen das Hauptkonzept an erster Stelle steht, umfasst. Der Name des F-Moleküls entspricht dem definierten Konzept und wird zur Abgrenzung kursiv dargestellt. Die Reihenfolge der vorgestellten Moleküle ergibt sich aus dem Vorgehen zur Durchführunge einer FMEA. Zu Beginn der Durchführung wird die Systemstruktur, anschließend die Funktionsstruktur und die Fehlfunktionsstruktur und abschließend wird die Maßnahmenstruktur entwickelt.[845]

Zusätzlich werden die wichtigsten Inferenzregeln zum jeweiligen Hauptkonzept ebenfalls aufgestellt und erläutert.

7.3.3.1 F-Molekül *Fmea*

Aus einem FMEA-Formular nach VDA´96 werden im Kopfbereich[846] die Instanz des betrachteten Gesamtsystems (`Typ/Modell/Fertigung/Charge`) und eine identifizierendes Konzept (`FMEA-Nr.`), dessen Instanzen zur Identifizierung der jeweils betroffenen FMEA-Durchführung dienen, für die FMEA-Ontologie entnommen. Hierzu werden jeweils Instanzen vom Typ STRING verwendet.[847] Ferner wird berücksichtigt, ob es sich um eine System-FMEA Produkt oder System-FMEA Prozess handelt. Für die Festlegung wird hier mit einem booleschen Wert[848] gearbeitet. Dabei entspricht der Wert `true` einer `system_fmea_produkt` und der Wert `false` entspricht einer `system_fmea_prozess`.[849] Das `datum` des Erstellzeitpunkts wird repräsentiert und der/die `verantwortungstraeger` wird/werden als Konzept festgelegt.

845) Siehe zur detaillierten Darstellung des Ablaufs zur Durchführung einer FMEA Kapitel 3.2, S. 57 ff., und insbesondere Kapitel 3.3, S. 62 ff.

846) Alternativ wird in dieser Arbeit auch synonym der Begriff „Header“ aus dem englischen Sprachraum verwendet.

847) Für die Relation `hat_fmea_nr` wird hier auf den Datentyp STRING zurückgegriffen, um auch „zahlenähnliche“ Klassifizierungen vornehmen zu können, wie sie bspw. im Anhang für die implementierten FMEAs Berücksichtigung finden.

848) Boolesche Werte sind *true* und *false.*

849) Diese indirekte Herangehensweise reicht an dieser Stelle aus, um die Kodierung nicht unnötig zu „überfrachten“, da von einer „geschlossenen Welt“ gemäß VDA ausgegangen werden kann, in der lediglich ein Produkt oder ein Prozess untersucht wird. Alternativ wäre eine erweiterte Modellierung denkbar, in der zusätzlich eine Relation `ist_system_fmea_prozess=>BOOLEAN` expliziert wird. Dies würde jedoch die Modellierung einer Integritätsregel notwendig machen, die sicherstellt, dass beide Relationen nie gleichzeitig erfüllt sind.

Die Angaben entsprechen den Stammdaten gemäß System-FMEA-Formblatt (siehe Abbildung 53) und werden in Form eines F-Moleküls für das Konzept fmea modelliert:

```
fmea[ist_system_fmea_produkt=>BOOLEAN;
     hat_fmea_nr=>STRING;
     betrachtet_system_von_typ_modell_fertigung_charge=>STRING;
     hat_verantwortung=>>verantwortungstraeger;
     wurde_erstellt=>datum;
     untersucht_systemelement=>systemelement].
```

Zusätzlich zu den Stammdaten wird die Betrachtung des untersuchten Systemelements mit der Relation untersucht_systemelement mit dem Konzept fmea verbunden. Diese Relation führt unmittelbar zum nächsten betrachteten Hauptkonzept systemelement. Eine Instanz einer fmea betrachtet gemäß den üblichen Vorgaben jeweils ein systemelement, das in dem betreffenden Feld des Formulars anzugeben ist.[850] Deshalb wird an dieser Stelle pro fmea nur eine einwertige Relation untersucht_systemelement festgelegt.

Fehler-Möglichkeits- und Einfluss-Analyse
☐ System-FMEA Produkt ☐ System-FMEA Prozess
FMEA-Nr.:

Typ/Modell/Fertigung/Charge: | Sach-Nr: Änderungsstand: | Verantw. Firma: | Abt.: Datum:
System-Nr./Systemelement: Funktion/Aufgabe: | Sach-Nr: Änderungsstand: | Verantw. Firma: | Abt.: Datum:

Mögliche Fehlerfolgen	B	Möglicher Fehler	Mögliche Fehlerursachen	Vermeidungs-maßnahmen	A	Entdeckungs-maßnahmen	E	RPZ	V/T

B: Bewertungszahl für die Bedeutung
V: Verantwortlichkeit
A: Bewertungszahl für die Auftretenswahrscheinlichkeit
T: Termin für die Erledigung
E: Bewertungszahl für die Entdeckungswahrscheinlichkeit
Risikoprioritätszahl RPZ = B*A*E

Abbildung 53: Stammdaten eines System-FMEA-Formblatts nach VDA'96

850) Siehe hierzu auch die Ausführungen in Kapitel 3.2.3, S. 59 ff., und Kapitel 3.3.2, S. 63 ff.

7.3.3.2 F-Molekül *Systemelement*

Untersuchte Systemelemente werden als Instanz des Konzepts `systemelement` repräsentiert. Die identifizierten Systemelemente, die sowohl Bestandteile eines komplexeren Systems als auch selbständige Systeme sein können, werden in einer Systemstruktur angelegt, die sich aus der Verknüpfung der identifizierten Systemelemente ergibt. Es wird das folgende F-Molekül angelegt:

```
systemelement[hat_systemelementbezeichnung=>STRING;
              ist_teil_von_systemelement=>>systemelement;
              ist_zusammengesetzt_aus_systemelement=>>systemelement;
              hat_schnittstelle_mit=>>systemelement;
              hat_funktion=>>funktion;
              wird_untersucht_von_fmea=>fmea].
```

Für jedes Systemelement wird eine Bezeichnung (`hat_systemelementbezeichnung`) in Form eines Strings hinterlegt. Die Beziehung zu anderen Systemelementen wird in der Ontologie über eine hierarchische Struktur, in der sich ein „oberes“ Systemelement aus mehreren „unteren“ Systemelementen zusammensetzt, dargestellt. Hierzu wird die Relation `ist_teil_von_systemelement` modelliert, um tatsächliche Verbindungen eines Systemelements zu seinem übergeordneten Systemelement darzustellen. In umgekehrter Richtung kann ein Systemelement verschiedene weitere Systemelemente als untergeordnete Systemelemente zugewiesen bekommen. Dies wird mittels der Relation `ist_zusammengesetzt_aus_systemelement` ermöglicht.

Ergeben sich innerhalb der hierarchischen Struktur Schnittstellen zu anderen Systemelementen, die nicht unmittelbares Ober- oder Unterkonzept sind, so wird dieser Sachverhalt über die Relation `hat_schnittstelle_mit` berücksichtigt.[851] Über die Relation `hat_funktion` wird analog zur Modellierung des F-Moleküls `fmea` die Beziehung zum nächsten betrachteten Hauptkonzept `funktion` hergestellt. Über diese Relation werden einem Systemelement Funktionen zugeordnet. Die Relation `wird_untersucht_von_fmea` stellt im Gegensatz zur vorgenannten Relation, die das nächste zu betrachtende Hauptkonzept verbindet, die Verbindung zum vorhergehend betrachteten Hauptkonzept her.

Die beiden genannten Relationen `wird_untersucht_von_fmea` und `untersucht_systemelement` verhalten sich symmetrisch zueinander. Mit Hilfe einer Inferenzregel wird dieses Symmetrieverhalten implementiert:

```
FORALL X,Y  X[untersucht_systemelement->Y] <->
            Y[wird_untersucht_von_fmea->X].
```

851) Siehe hierzu auch die Ausführungen in Kapitel 3.1.4.1.1, Seite 47 ff.

Wenn eine FMEA X ein Systemelement Y untersucht, dann gilt in gleicher Weise gleichzeitig, dass das Systemelement Y von der FMEA X untersucht wird. Gleiches gilt für die beiden genannten Relationen `ist_teil_von_systemelement` und `ist_zusammengesetzt_aus_systemelement`, für die ebenfalls eine Inferenzregel für ihr Symmetrieverhalten in der Ontologie implementiert wurde:

```
FORALL X,Y  X[ist_teil_von_systemelement->>Y] <->
            Y[ist_zusammengesetzt_aus_systemelement->>X].
```

Wenn ein Systemelement X ein untergeordnetes Systemelement eines Systemelements Y ist, dann ist das Systemelement Y zusammengesetzt aus dem Systemelement X und möglicherweise weiteren Systemelementen.

7.3.3.3 F-Molekül *Funktion*

Jedem Systemelement wird mindestens eine Funktion, die das Systemelement erfüllen soll, zugeordnet. Das F-Molekül *Funktion* wird benötigt, um die Funktionen und die Funktionsstruktur einer FMEA darzustellen. Für jede Funktion in der Ontologie wird eine Bezeichnung über die Relation `hat_funktionsbezeichnung` in Form eines Strings zum Konzept `funktion` angelegt, und es wird angegeben, welchem Systemelement die Funktion zugeordnet ist (`ist_funktion_von_systemelement`).

Die Beziehung zu anderen Funktionen wird in der Ontologie über eine hierarchische Struktur, in der sich eine „obere" Funktion aus mehreren „unteren" Funktionen zusammensetzt, dargestellt. Hierzu wird die Relation `ist_teil_von_funktion` modelliert, um tatsächliche Verbindungen einer Funktion zu seiner übergeordneten Funktion darzustellen. In umgekehrter Richtung kann eine Funktion verschiedene weitere Funktionen als untergeordnete Funktionen zugewiesen bekommen, dies wird mittels der Relation `ist_zusammengesetzt_aus_funktion` ermöglicht. Die genannten Relationen zur Hierarchisierung von Funktionen beinhalten, dass eine Funktion dann generiert wird, wenn ein Systemelement diese Funktion unabhängig von seiner Anordnung im Gesamtsystem eigenständig ausführt[852], nicht jedoch, wenn sie lediglich als Teil einer anderen Funktion desselben Systemelements angesehen werden muss. Es ergibt sich folgende Kodierung für das F-Molekül *Funktion*:

```
funktion[hat_funktionsbezeichnung=>STRING;
         ist_funktion_von_systemelement=>>systemelement;
         ist_teil_von_funktion=>>funktion;
         ist_zusammengesetzt_aus_funktion=>>funktion;
         hat_fehlermoeglichkeit=>>fehler].
```

Über die Relation `hat_fehlermoeglichkeit` werden einer Funktion beliebig viele mögliche Fehler zugeordnet. Die Zuordnung von Fehlern führt zu dem im nächsten Unterka-

852) Vgl. VDA-Band 4 (2003), S. 17.

pitel beschriebenen F-Molekül *Fehler*. Zunächst sind jedoch noch zwei Symmetrieregeln zu berücksichtigen.

Die beiden Relationen `ist_teil_von_funktion` und `ist_zusammengesetzt_aus_funktion` verhalten sich symmetrisch zueinander. Mit Hilfe einer Inferenzregel wird dieses Symmetrieverhalten implementiert:

```
FORALL X,Y  X[ist_teil_von_funktion->>Y] <->
            Y[ist_zusammengesetzt_aus_funktion->>X].
```

Wenn eine Funktion Y zusammengesetzt ist aus einer Funktion X, dann gilt in gleicher Weise, dass die Funktion X Teil der Funktion Y ist. Gleiches gilt für die beiden genannten Relationen `hat_funktion` und `ist_funktion_von_systemelement`, für die ebenfalls eine Inferenzregel für ihr Symmetrieverhalten in der Ontologie implementiert wurde:

```
FORALL X,Y  X[hat_funktion->>Y] <->
            Y[ist_funktion_von_systemelement->>X].
```

Wenn eine Funktion Y einem Systemelement X zugeordnet wurde, dann hat das Systemelement X die Funktion Y.

7.3.3.4 F-Molekül *Fehler*

Für die Funktionen sind bei der Durchführung einer FMEA Fehler zu identifizieren. Für jeden Fehler in der Ontologie wird eine Bezeichnung über die Relation `hat_fehlerbezeichnung` in Form eines Strings angelegt und es wird angegeben, welche Funktionserfüllung der Fehler verhindert (`verhindert_funktion`). Die Beziehungen zu weiteren Fehlern werden in der Ontologie über eine netzartige Struktur[853], in der sich ein Fehler aus mehreren Fehlerursachen zusammensetzen und mehrere Folgefehler haben kann, dargestellt. Die Relation `verursacht_fehler` wird verwendet, um mögliche Verbindungen eines Fehlers zu seinen Fehlerfolgen darzustellen. In umgekehrter Richtung kann ein Fehler verschiedene Fehlerursachen besitzen, die mittels der Relation `hat_ursache` dargestellt werden. Es wird somit nur ein Konzept `fehler` verwendet, das die FMEA-Konstrukte *Fehler, Fehlerursache* und *Fehlerfolge* vereinigt.[854]

853) Siehe hierzu auch Abbildung 8, S. 45.

854) An dieser Stelle sei nochmals daran erinnert, dass ein Fehler in Abhängigkeit von der Betrachtungsweise zugleich auch eine Fehlerursache oder eine Fehlerfolge für bspw. die Funktion eines anderen Systemelements sein kann. Siehe hierzu auch Kapitel 3.3.4, S. 67 f. Die hier vorgeschlagene Modellierung unterstützt diese Möglichkeit. Im Gegensatz hierzu würde bspw. eine Modellierung, die die Konzepte `fehlerursache` und `fehlerfolge` als Unterkonzepte von `fehler` betrachten würde, es notwendig machen, zahlreiche weitere Relationen einzuführen, um die jeweiligen dreistelligen Relationen aus Fehler, Fehlerfolge und Fehlerursache zu allen Systemelementen und den zugehörigen Funktionen jeweils eindeutig zuordnen zu können. Nach Ansicht des Verfassers stellt die vorliegende Modellierung hier den kürzesten und auch „elegantesten“ Ansatz dar.

Es ergibt sich folgende Kodierung für das F-Molekül *Fehler*:

```
fehler[hat_fehlerbezeichnung=>STRING;
       verhindert_funktion=>>funktion;
       verursacht=>>fehler;
       hat_ursache=>>fehler;
       hat_gesamtrpz=>>INTEGER;
       hat_fehlertupel=>>fehlertupel].
```

Die beiden Relationen `verursacht` und `hat_ursache` sind symmetrisch. Mit Hilfe einer Inferenzregel wird diese Symmetrie implementiert:

```
FORALL X,Y  X[hat_ursache->>Y] <->
            Y[verursacht->>X].
```

Wenn ein Fehler Y eine Fehlerfolge X verursacht, dann hat der Fehler X die Ursache Y. Gleiches gilt für die beiden genannten Relationen `hat_fehlermoeglichkeit` und `verhindert_funktion`, für die ebenfalls eine Inferenzregel für ihr Symmetrieverhalten in der Ontologie implementiert wurde:

```
FORALL X,Y  X[hat_fehlermoeglichkeit->>Y] <->
            Y[verhindert_funktion->>X].
```

Die Relation `hat_gesamtrpz` ist für die Repräsentation des Wissens einer FMEA nicht notwendig. Ein ganzzahliger Wert vom Typ Integer gibt dabei die Summe aller Risikoprioritätszahlen, die zu einem Fehler ermittelt werden können, wieder. Sie wird bei der Wiederverwendung des Wissens einer FMEA angewendet, um die Benutzerfreundlichkeit zu erhöhen.[855]

Über die Relation `hat_fehlertupel` werden einem Fehler beliebig viele mögliche Fehlertupel zugeordnet. Dies führt zum nächsten Unterkapitel, in dem das F-Molekül *Fehlertupel* beschrieben wird.

7.3.3.5 F-Molekül *Fehlertupel*

Eine Zeile aus einem FMEA-Formblatt (der betrieblichen Realität) entspricht einer Instanz des F-Moleküls *Fehler* (s.o.) in Verbindung mit einer Instanz des F-Moleküls *Fehlertupel* (siehe auch Abbildung 54, S. 239).

In der Risikobewertung werden für jede Kombination von Fehler, Fehlerursache und Fehlerfolge, die zuvor im F-Molekül *Fehler* festgelegt wurde, Gewichtungen hinsichtlich Bedeutung, Auftrittswahrscheinlichkeit und Enteckungswahrscheinlichkeit vorgenommen und eine Risikoprioritätszahl ermittelt. Die Ermittlung der Gewichtungen bezieht sich vor allem auf ei-

855) Aufgrund ihrer nachgelagerten Bedeutung wird an dieser Stelle nicht weiter auf die Relation eingegangen. Der Einsatzzweck ergibt sich aus Kapitel 8.4.3, S. 274 ff.

nen bestimmten Zeitpunkt, zu dem festgelegte Maßnahmen bei der Risikoabschätzung in Betracht gezogen werden. Um diese Informationen einheitlich zusammenzufassen, wird das Konzept `fehlertupel` in die Ontologie aufgenommen.

Die Instanzen des F-Moleküls *Fehlertupel* beinhalten *direkt* oder *indirekt* (s.u.) das Wissen aus den Zeilen eines FMEA-Formblatts und zusätzlich noch die Information über den Zeitpunkt der Erstellung dieses Wissens.

Direkt enthalten sind über die Relationen `hat_bedeutung`, `hat_auftrittswahrscheinlichkeit` und `hat_entdeckungswahrscheinlichkeit` die Einzelfaktoren zur Ermittlung der Risikoprioritätszahl, die ebenfalls – über die Relation `hat_rpz` – einer Instanz des Konzepts `fehlertupel` direkt dem Fehlertupel zugeordnet wird. Um diese Einzelfaktoren ermitteln zu können, müssen Vermeidungs- und Entdeckungsmaßnahmen direkt dem Fehlertupel zugeordnet werden (Relation: `hat_massnahme`).[856] Weil diese Informationen vom Betrachtungszeitpunkt (genauer: vom Zeitpunkt der Durchführung einer FMEA) abhängig sind, muss über die Relation `datum_des_fehlertupels` auch noch ein Datum berücksichtigt werden. Hinzu kommt die Möglichkeit, mindestens eine verantwortliche Person (samt mindestens eines verantwortlichen Unternehmens, für das die Person arbeitet) für die festgelegten Maßnahmen einschließlich eines Erledigungstermins definieren zu können. Hierzu werden die Relationen `ist_verantwortlich` und `hat_termin` eingeführt.

Über die Relation `ist_fehlerursache_zugeordnet` wird das Konzept `fehlertupel` mit dem Konzept `fehler` verbunden.[857] Weil bei der Durchführung einer FMEA in der Phase der Optimierung gegebenenfalls ein „neues" Fehlertupel in das Formblatt eingetragen wird, ist es notwendig, dass die Relation `ist_fehlerursache_zugeordnet` mehrwertig ausgeprägt wird. Durch diese mehrwertige Relation und die eindeutige Festlegung von Instanzen zum Datum des Fehlertupels wird es möglich in der FMEA-Ontologie, Veränderungsstände zu berücksichtigen.

856) Siehe hierzu auch das nachfolgende Unterkapitel zum F-Molekül *Maßnahme*.

857) Streng genommen wird das eigentliche Tupel aus Fehler, Fehlerfolge und Fehlerursache bereits mit Hilfe des F-Moleküls *Fehler* festgelegt, so dass der Begriff „Fehlertupel" für das Konzept `fehlertupel` zu Anfang irreführend erscheint. Weil jedoch das F-Molekül *Fehler* in der Hauptsache der Festlegung eines Fehlers und nicht der Festlegung des besagten Tupels dient, soll dieser Umstand entsprechend begrifflich hervorgehoben werden, indem lediglich der Begriff „Fehler" verwendet wird. Ferner lässt sich für das F-Molekül *Fehlertupel* seine Namensgebung dahingehend begründen, dass die Maßnahmen und vor allem die Wahrscheinlichkeiten/Bedeutungen (E, A und B) notwendig für die Vollständigkeit der Angabe des FMEA-Wissens aus den Spalten des Formblatts sind. Im vorliegenden Fall werden - mit anderen Worten - die Angaben zu Maßnahmen und Wahrscheinlichkeiten/Bedeutungen als konstituierend für ein solches Tupel angesehen. Zur Verdeutlichung ließe sich auch vom „Stand eines Fehlertupels" ausgehen, d. h. das F-Molekül *Fehlertupel* legt einen zu einem bestimmten Datum gültigen Ausdruck einer Spalte des FMEA-Formblatts als Instanz fest. Der Verfasser folgt jedoch der Politik, Konzepte nach Möglichkeit mit Hilfe eines Begriffs zu bezeichnen, so dass statt `stand_des_ fehlertupels` das Konzept `fehlertupel` Verwendung findet.

	Fehler-Möglichkeits- und Einfluß-Analyse ☐ System-FMEA Produkt ☐ System-FMEA Prozeß			FMEA-Nr.:
Typ/Modell/Fertigung/Charge:		Sach-Nr: Änderungsstand:	Verantw. Firma:	Abt.: Datum:
System-Nr./Systemelement: Funktion/Aufgabe:		Sach-Nr: Änderungsstand:	Verantw. Firma:	Abt.: Datum:

Mögliche Fehlerfolgen	B	Möglicher Fehler	Mögliche Fehlerursachen	Vermeidungs-maßnahmen	A	Entdeckungs-maßnahmen	E	RPZ	V/T

B: Bewertungszahl für die Bedeutung
V: Verantwortlichkeit
A: Bewertungszahl für die Auftretenswahrscheinlichkeit
T: Termin für die Erledigung
E: Bewertungszahl für die Entdeckungswahrscheinlichkeit
Risikoprioritätszahl RPZ = B*A*E

Abbildung 54: Zusammensetzung des Konzepts `fehlertupel`

Die Relation `ist_fehlerursache_zugeordnet` führt zum *indirekt* enthaltenen Wissen zu möglichen Fehlern und Fehlerfolgen einer Fehlerursache. Das heißt, ein Fehlertupel bezieht sich immer auf eine bestimmte Fehlerursache in der Wissensbasis. Hierdurch lassen sich Instanzen des Konzepts `fehlertupel` eindeutig referenzieren, weil ein Fehler mehrere Fehlerursachen und Fehlerfolgen haben kann.[858] Über das bereits erläuterte Konzept `fehler` können diese Informationen ermittelt werden.

Es ergibt sich folgende Kodierung in F-Logic des F-Moleküls *Fehlertupel*:

```
fehlertupel[hat_bedeutung=>INTEGER;
            hat_auftrittswahrscheinlichkeit=>INTEGER;
            hat_entdeckungswahrscheinlichkeit=>INTEGER;
            hat_rpz=>INTEGER;
            ist_fehlerursache_zugeordnet=>>fehler;
            hat_massnahme=>>massnahme;
            ist_verantwortlich=>>verantwortungstraeger;
            hat_termin=>datum;
            datum_des_fehlertupels=>datum].
```

Aus dem Umstand, dass einem Fehlertupel Y eine Fehlerursache X zugeordnet ist, lässt sich ableiten, dass zur Fehlerursache X das Fehlertupel Y gehört (Relation: `hat_fehlertupel`)

858) Siehe hierzu auch die Ausführungen in Kapitel 3.1.4.1.3.2, S. 50.

und vice versa.[859] Dieser Sachverhalt wird mittels der folgenden Regel in der Ontologie festgehalten:

```
FORALL X,Y  X[hat_fehlertupel->>Y] <->
            Y[ist_fehlerursache_zugeordnet->>X].
```

Um die Relation `hat_massnahme`, die auf das Konzept `massnahme` referenziert, einsetzen zu können, muss selbiges Konzept noch in die Ontologie eingeführt werden. Dies erfolgt im nächsten Unterkapitel mit Hilfe des F-Moleküls *Maßnahme*. Analog gelten die Ausführungen zu Symmetrieregeln auch für die Relation `ist_verantwortlich` und das Konzept `verantwortungstraeger`, die im darauf folgenden Unterkapitel eingeführt werden.

7.3.3.6 F-Molekül *Maßnahme*

Für die Festlegung von Maßnahmen auf der Instanzenebene wird das Konzept `massnahme` gebraucht. Bezüglich einer Fehlerursache wird dabei in Entdeckungs- und Vermeidungsmaßnahmen unterschieden.[860] Insbesondere, wenn die ermittelte Risikoprioritätszahl in den Augen der Ersteller einer FMEA zu „hoch" ausfällt, müssen Maßnahmen ermittelt werden, die bei erfolgreicher Umsetzung eine „niedrigere" Prioritätszahl nach sich ziehen. Für eine Maßnahme wird in der Ontologie über die Relation `hat_bezeichnung` ermöglicht, die Maßnahme in Form eines Strings zu benennen. Über die Relation `wird_angewendet` wird eine Maßnahme einem Fehlertupel (s. o.) zugewiesen:

```
massnahme[hat_bezeichnung=>STRING;
          wird_angewendet=>>fehlertupel].
```

Die Unterscheidung von Entdeckungs- und Vermeidungsmaßnahmen erfolgt in der Ontologie über die beiden Unterkonzepte `entdeckungsmassnahme` und `vermeidungsmassnahme`, die beide vom Oberkonzept `massnahme` die zugehörigen Relationen erben (siehe auch oben):[861]

```
entdeckungsmassnahme::massnahme
vermeidungsmassnahme::massnahme
```

---

859) Siehe hierzu auch die Ausführungen zum F-Molekül *Fehler* im voranstehenden Kapitel.

860) Siehe hierzu auch die Ausführungen in Kapitel 3.1.4.1.4, S. 50 f.

861) An dieser Stelle bleibt die vorgestellte FMEA-Ontologie unvollständig, weil sich für die Umsetzung in den Prototyp (siehe Kapitel 1, S. 252 ff.) eine weitere Festlegung der formalen Semantik als nicht notwendig herausgestellt hat. Um die formale Semantik der FMEA-Ontologie zu vervollständigen, sollte sich ein inhaltlicher Unterschied zwischen den beiden Unterkonzepten `entdeckungsmassnahme` und `vermeidungsmassnahme` finden lassen. Bspw. wäre es möglich, die Integer-Werte für die Auftrittswahrscheinlichkeit und die Entdeckungswahrscheinlichkeit direkt mit zwei verschiedenen Relationen den beiden Unterkonzepten zuzuordnen.

Wenn eine Maßnahme Y auf ein Fehlertupel X angewendet wird, dann hat im Umkehrschluss das Fehlertupel X die Maßnahme Y. Diese Symmetriebedingung wird über die folgende Regel abgebildet:

```
FORALL X,Y  X[hat_massnahme->>Y] <->
            Y[wird_angewendet->>X].
```

7.3.3.7 F-Atom *Verantwortungsträger*

Das Konzept `fmea` und das Konzept `fehlertupel` werden über die Relationen `hat_verantwortung` bzw. `ist_verantwortlich` mit dem Konzept `verantwortungstraeger` verbunden. Das Konzept `verantwortungstraeger` kann in die beiden Konzepte `unternehmen` und `person` spezialisiert werden.[862] Für eine Instanz des Konzepts `verantwortungstraeger` wird jeweils ein Name über die Relation `hat_name` als String angelegt. Zusätzlich wird für das Unterkonzept `person` die Relation `arbeitet_fuer` angelegt. Diese Relation erlaubt es, eine Person einem Unternehmen zuzuordnen.[863] Es ergeben sich zu der bereits aufgeführten Kodierung folgende Ergänzungen:

```
verantwortungstraeger[hat_name=>STRING;
                      ist_verantwortlich_fuer_fehlertupel=>>fehlertupel].
person[arbeitet_fuer=>>unternehmen].
```

Wenn ein Verantwortungsträger Y verantwortlich für ein Fehlertupel X ist, dann ist für das Fehlertupel X der Verantwortungsträger Y verantwortlich. Diese Symmetriebedingung der beiden Relationen `ist_verantwortlich_fuer_fehlertupel` und `ist_verantwortlich` wird mittels der folgenden Regel in der Ontologie berücksichtigt:

```
FORALL X,Y  X[ist_verantwortlich->>Y] <->
            Y[ist_verantwortlich_fuer_fehlertupel->>X].
```

862) Siehe hierzu auch Fn. 844.

863) Um den Fokus der Zielsetzung dieser Arbeit nicht aus den Augen zu verlieren, wird in der vorliegenden FMEA-Ontologie auf eine weitergehende Modellierung an dieser Stelle verzichtet, obwohl es bspw. unmittelbar einleuchtet, Adress- und Telefonverbindungen für einen Verantwortungsträger als mögliche Konzepte zu realisieren, um den Anwendungskomfort von FMEA-Durchführungen, die auf der FMEA-Ontologie aufbauen, in der betrieblichen Praxis zu erhöhen.

7.3.3.8 F-Molekül *Datum*

Den Konzepten `fmea` und `fehlertupel` müssen Datumsangaben zugeordnet werden. Hierzu wird das F-Atom *Datum* eingeführt, das aus den drei Relationen `tag`, `monat` und `jahr` besteht. Die drei Relationen verweisen jeweils auf einen möglichen Integer-Wert:

```
datum[tag=>INTEGER;
      monat=>INTEGER;
      jahr=>INTEGER].
```

7.3.4 Gesamtdarstellung der Meta-Ebene

Die Abbildung 55 auf der nächsten Seite verdeutlicht noch einmal zusammenfassend die Hauptkonzepte der Meta-Ebene der FMEA-Ontologie.

Die Notation erfolgt in der Abbildung in Anlehnung an die Darstellung von Klassen-Diagrammen gemäß der Unified Modeling Language.[864]

Die Relationen, die die Hauptkonzepte der FMEA mit primitiven Datentypen verknüpfen, sind jeweils unterhalb der Konzeptnamen im Kasten aufgelistet. Die primitiven Datentypen sind bereits durch die Meta-Meta-Ebene vorgegeben, d. h. Ontobroker erkennt die Semantik dieser Ausdrücke, ohne dass eine weitere Definition erforderlich wird. Die weiteren Relationen werden durch Kanten repräsentiert, die in Abhängigkeit von der Leserichtung benannt wurden. Hier gibt ein < oder > die jeweilige Wirkungsrichtung der Relation an. Bspw. wenn es für das Konzept `fmea` eine Relation `wurde_erstellt>` gibt, die auf das Konzept `datum` verweist, dann wurde eine FMEA zu einem bestimmten Datum erstellt und nicht ein Datum zu einer bestimmten FMEA erstellt. In F- Logic ergibt sich hierzu die Kodierung: `fmea[wurde_erstellt=>datum]`. Stehen zwei Bezeichnungen an einer Kante, so verhalten sich die Relationen symmetrisch. Zusätzlich sind zu jeder Relation die Kardinalitäten, die die mögliche Anzahl von Instanzen an einer Relation angeben, notiert. Die einem ausgehenden Konzept gegenüberliegende Zahl legt fest, wie oft Instanzen des verbundenen Konzepts an der festgelegten Relation beteiligt sind. Es werden insgesamt 4 Kardinalitäten unterschieden, die sich mit Hilfe der Ausprägungen „0", „1" und „*" darstellen lassen. Während die Ausprägungnen „0" und „1" selbsterklärend sind, definiert „*", dass beliebig viele aber stets endlich viele Instanzen dem verbundenen Konzept zugeordnet werden können (n>0). Es finden sich die Kardinalitäten 0..1, 1..1, 0..* und 1..*. Bspw. lässt sich für die Relation `verhindert_funktion` die Kardinalität 1..* ermitteln. Diese Kardinalität gibt an, dass zu jeder Instanz von `fehler` mindestens eine (und höchstens endlich viele) Instanz(en) von `funktion` existieren muss (müssen).

Unterkonzeptbeziehungen werden mittels Kanten, die Pfeilspitzen haben, dargestellt.

864) Vgl. OMG (2003), S. 3-1 ff.

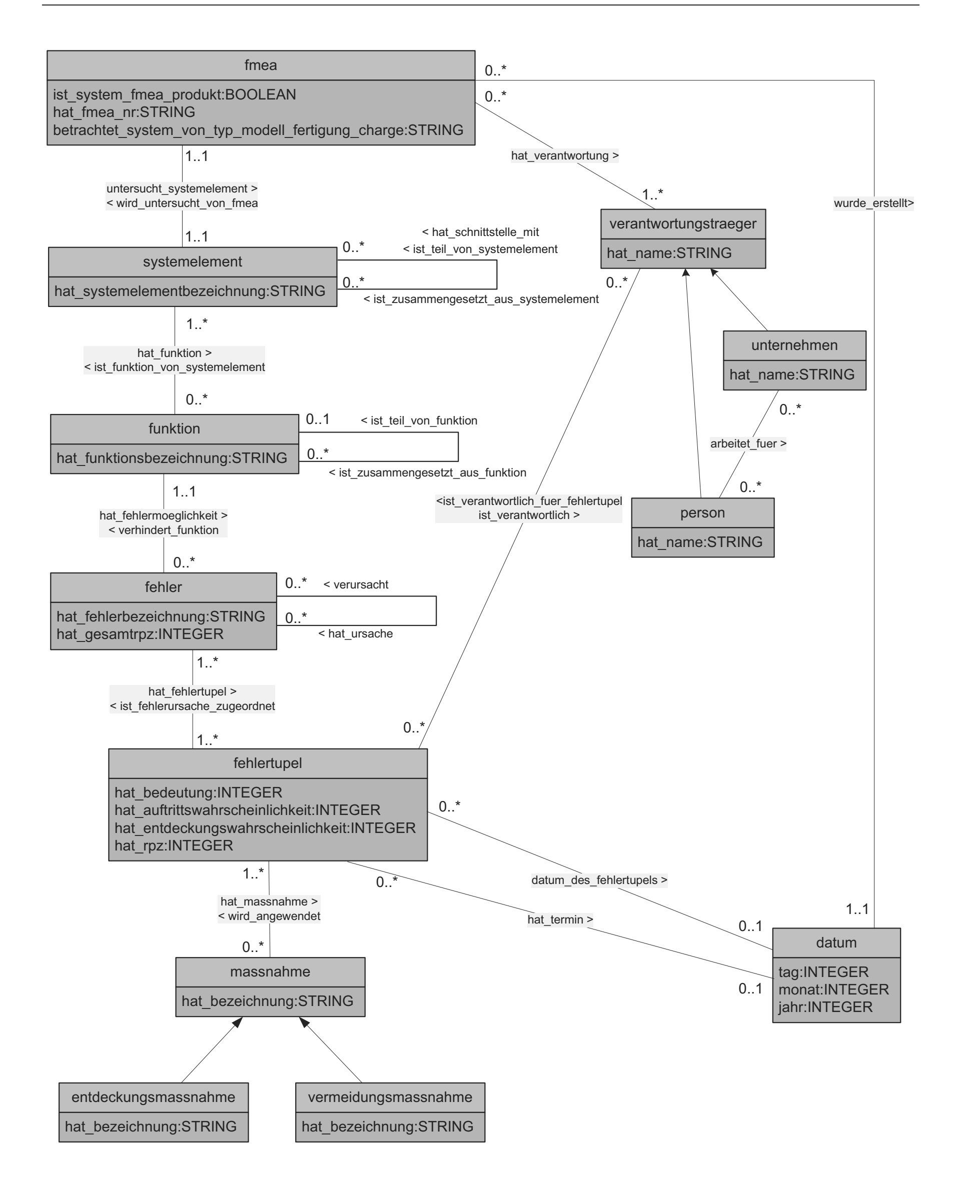


Abbildung 55: Hauptkonzepte der Meta-Ebene

7.4 Klassen-Ebene

7.4.1 Beschreibung der Ebene

Mit dem „Herabsteigen“ von der Meta-Meta-Ebene bis zur Instanzen-Ebene nimmt der Grad der „Generalisierung“ der modellierten Entitäten innerhalb einer Domäne ab. Somit lässt sich sagen, dass ein fließender Übergang von „relativer Neutralität“ (den Modellierungsprimitiven der Wissensrepräsentationssprache) zu den Subjekten des betrieblichen Einsatzfeldes der FMEA-Ontologie stattfindet. Während sich die beiden voranstehenden Ebenen noch relativ „neutral“ anhand allgemeingültiger Grundsätze (Normen, Richtlinien) und Vorgaben (Spezifikation von F-Logic, Objektorientierte Programmierung) herleiten ließen, beziehen sich bereits in der Klassen-Ebene die zu modellierenden Konzepte stark auf den gemeinsam wahrgenommenen Realitätsausschnitt der Mitglieder eines Unternehmens, in dem die FMEA-Ontologie eingesetzt werden soll.[865]

Die Klassen-Ebene wird genutzt, um wichtige Konzepte der Meta-Ebene näher zu spezifizieren. Diese näheren Spezifikationen dienen der Unterscheidung von Instanzen. Durch sie wird ermöglicht, dass eine Ähnlichkeit von Instanzen bestimmbar wird.[866] Weil die nähere Spezifikation oftmals über taxonomische Relationen vorgenommen wird, kann an dieser Stelle alternativ auch von einer „Taxonomie-Ebene“ gesprochen werden.

Für eine detaillierte Beschreibung bieten sich insbesondere die beiden Konzepte `funktion` und `systemelement` der Meta-Ebene an, weil mit ihrer Hilfe für neu zu erstellende FMEAs Vorschläge hinsichtlich möglicher Systemelemente, Funktionen und Fehlermöglichkeiten abgeleitet werden können.[867] Dieses abgeleitete Wissen soll dem Benutzer eines ontologiebasierten FMEA-Systems zur Verfügung gestellt werden, um die Arbeitsabläufe wesentlich zu erleichtern. Des Weiteren ermöglicht eine „Strukturierung“ der möglichen Unterkonzepte dieser beiden Konzepte eine einheitliche Repräsentation von FMEAs, die in einer Wissensbasis abgelegt werden. Speziell wird durch die Strukturierung die Wiederverwendung des angelegten Wissens gefördert.

Auch für das Konzept `massnahme` bietet sich unter Umständen eine Taxonomie als Vorgabe möglicher Unterkonzepte zur Strukturierung des Wissens an, sofern die FMEA-Ontologie in Bereichen eingesetzt wird, in denen zahlreiche Maßnahmen bekannt sind und wiederholt verwendet werden. Für die vorliegende Untersuchung wird das Fallbeispiel der Karl Schumacher Maschinenbau GmbH herangezogen. Die Karl Schumacher Maschinenbau GmbH unterschei-

865) Der Begriff „neutral“ bezieht sich hier vereinfacht auf die Abhängigkeit eines Realitätsauschnitts von der gemeinsamen Wahrnehmung. Wird ein solcher Ausschnitt modelliert, so werden Hilfsmittel (wie etwa eine Repräsentationssprache), die a priori existieren, als neutral gegenüber der aktiven Leistung der Wahrnehmung bei der Modellierung angesehen. In diesem Sinne nimmt mit der verstärkten Einbeziehung des wahrgenommenen Realitätsausschnitts durch die Entwickler der Grad der Subjektivität zu, d. h. Instanzen als Bestandteile der untersten Ebene gelten als „weniger neutral“ als Kategorien von Instanzen.

866) Auf diesen Zweck wird später näher eingegangen. Siehe hierzu die Kapitel 8.3.2, S. 257 ff., und 8.3.3, S. 260 ff.

867) Siehe hierzu die voranstehende Fn.

det lediglich – wie die Vorgabe VDA´96 – zwischen Vermeidungs- und Entdeckungsmaßnahmen.

Für die Konzepte `systemelement` und `funktion` werden exemplarisch in die FMEA-Ontologie taxonomische Strukturen eingeführt. Mangels Masse wird auf eine taxonomische Struktur für das (Ober-)Konzept `massnahme` verzichtet. Eine taxonomische Struktur besteht aus einem Oberkonzept, der Menge aller seiner Unterkonzepte und den dazugehörigen Abstraktionsrelationen.[868] Als Argumentationshilfen dienen diese taxonomischen Strukturen später der Verdeutlichung der Anwendbarkeit der FMEA-Ontologie in Kapitel 8.

7.4.2 Taxonomische Struktur *Systemelement*

Eine Instanz des Konzepts `systemelement` wird über die Relationen `ist_zusammengesetzt_aus` und `ist_teil_von` mit anderen Instanzen desselben Konzepts `systemelement` verbunden, d. h. es gibt eine Menge von Instanzen des Konzepts `systemelement` in der Wissensbasis, die über eine Menge von Instanziierungen (Beziehungen) der beiden vorgenannten Relationen miteinander verbunden sind. Diese Menge wird bei zunehmender Wissensbasis größer und damit weniger überschaubar. Um diese Menge von Instanzen zu „beherrschen“ (vor allem nachvollziehen zu können), wird eine Taxonomie eingeführt, die die Konzepte systematisiert.[869] Später (im folgenden Kapitel 8) werden mit Hilfe der Taxonomie Vorschläge für mögliche Systemelemente aus der Wissensbasis abgeleitet. Das oberste Element der eingeführten Taxonomie *Systemelement* (siehe Abbildung 56) wird durch das Konzept `systemelement` gebildet, das als Betrachtungseinheit aufgefasst wird.[870]

Die Taxonomie ermöglicht die Berücksichtigung einer Systemstruktur hinsichtlich der beiden Dimensionen für eine mögliche Betrachtungseinheit (*Betrachtungskriterium* und *Betrachtungsebene*).[871] Es wird zunächst das Konzept `systemelement` in die Unterkonzepte `einheit` und `system` ausdifferenziert.

---

868) Siehe hierzu auch die Ausführungen in Kapitel 4.3, S. 114 f.

869) Der Verfasser folgt hier der plausiblen Auffassung, dass eine nachvollziehbare Systematisierung das Auffinden von Objekten tendenziell erleichtert. Eine Systematisierung von Konzepten mittels einer Taxonomie führt in diesem Sinne zu einer „Beherrschung“ einer Instanzenmenge, d. h., über die Unterkonzeptbeziehungen lassen sich etwaige Instanzen überschauen und hinsichtlich ihrer Einordnung durch den Benutzer der Ontologie nachvollziehen (aus einem ungeordneten Haufen von Instanzen wird eine geordnete Menge von Instanzen).

870) Zum Begriff „Betrachtungseinheit“ siehe auch Kapitel 3.3.1, S. 62. Eine Betrachtungseinheit wird selbst nicht als Gliederungsmerkmal eines Ordnungsschemas aufgefasst. Die Betrachtungseinheit wird jedoch genutzt, um eine Betrachtungsebene festzulegen, der schließlich ein Ordnungsschema zugeordnet werden kann (vgl. DIN 40150:1979, Kap. 2, S. 1). Dies ermöglicht es, eine Wortbildung aus dem Begriff „System“ und der Endung „~element“ als oberstes Konzept einzuführen, obwohl in der Betrachtungsebene innerhalb des hierarchischen Aufbaus System und Element unterschieden werden. Aufgrund dieses „Kniffs“ wird es möglich, die übliche Terminologie für eine FMEA, die den Begriff „Systemelement“ verwendet, mit der im folgenden Text näher erläuterten Taxonomie zu verbinden, um eine systematische Einordnung von untersuchten „Systemelementen“ hinsichtlich Kriterium und Ebene zu ermöglichen.

871) Siehe hierzu auch die Ausführungen in Kapitel 3.3.2, S. 63 ff.

Dem Konzept `einheit` werden in dem Ausschnitt der zugehörigen Unterkonzepte der Abbildung 56 die Unterkonzepte `baueinheit`, `funktionseinheit` und `betriebseinheit` zugeordnet. Dem Konzept `funktionseinheit` werden beispielhaft die Unterkonzepte `bedieneinheit`, `feststelleinheit` und `transporteinheit` zugeordnet.[872] Während die Unterscheidung der Konzepte `einheit` und `system` allgemeingültigen Charakter für FMEAs hat, beginnt bereits auf der Ebene darunter das domänenspezifische Wissen des Unternehmens, das mit Hilfe der FMEA-Ontologie sein Wissen repräsentieren will. Das bedeutet, hier werden lediglich Beispiele für eine Gliederung angegeben. In der „realen" Umsetzung müssen an dieser Stelle die spezifischen Kriterien der Domäne, für die die Ontologie entwickelt wird, berücksichtigt werden.[873]

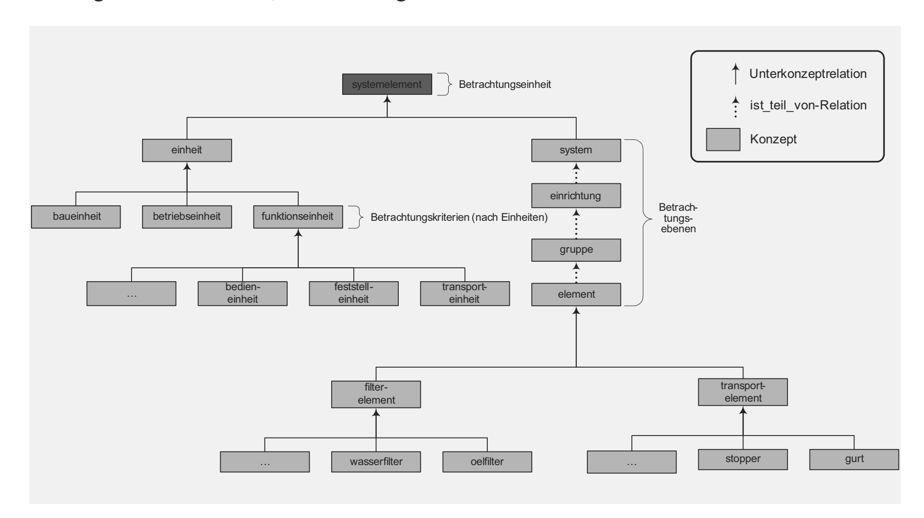


Abbildung 56: Ausschnitt aus der FMEA-Ontologie zum Konzept `systemelement`

Dem Konzept `system` wird über eine `ist_teil_von`-Relation (in der Abbildung als gestrichelte Kante dargestellt) das Konzept `einrichtung` zugeordnet. Dem Konzept `einrichtung` wird das Konzept `gruppe` mit derselben Relation zugeordnet. Dem Konzept `gruppe` wird wiederum mit derselben Relation das Konzept `element` zugeordnet. Ein Konzept `system` besteht im Regelfall aus mindestens einem Konzept `einrichtung`, ein Konzept `einrichtung` besteht mindestens aus einem Konzept `gruppe` und ein Konzept `gruppe` besteht mindestens aus einem Konzept `element`. Auch hier sind weitere Konzepte zur Differenzierung denkbar, die entsprechend der repräsentierten Domäne angepasst werden müssen. Lediglich die Konzepte `system` und `element` gelten als feststehend. Ein Konzept `system` stellt dabei immer die höchste Betrachtungsebene und das Konzept `element` immer die niedrigste Betrachtungsebene dar. Beispielhaft wurden im Ausschnitt der FMEA-

872) Die Beispielhaftigkeit dieser Unterkonzepte wird in der Abbildung 56, S. 246, durch eine leere Konzeptdarstellung, die mit drei Punkten (für eine beliebige Verwendung) versehen wurde, dargestellt.

873) Die hier gemachten beispielhaften Vorschläge für Konzepte entsprechen den Eigenarten der Domäne, die durch die Karl Schumacher Maschinenbau GmbH und deren Mitglieder (Angestellte, Arbeiter usw.) vorgegeben sind.

Ontologie der Abbildung 56 dem Konzept `element` mögliche Unterkonzepte zugeordnet, die wiederum Unterkonzepte besitzen können (Unterkonzept `filterelement` mit den Unterkonzepten `oelfilter` und `wasserfilter` sowie das Konzept `transportelement` mit den Unterkonzepten `stopper` und `gurt`).

7.4.3 Taxonomische Struktur *Funktion*

Um an dieser Stelle eine Allgemeingültigkeit für die gemachten Annahmen und Ergebnisse reklamieren zu können, wird eine Gliederung der Funktionen nach grammatikalischen Gesichtspunkten vorgenommen. Die Grammatik der deutschen Sprache wird dabei als ein verbindendes Element aller denkmöglichen Realitätsausschnitte verstanden.[874] Zusätzlich wird davon ausgegangen, dass alle in der Literatur genannten Begriffe zur Darstellung technischer Funktionen *Verben* sind oder zumindest Verben enthalten.[875] Für die Identifikation von Verben wird auf die Arbeit von Birkhofer, der die Funktionen technischer Systeme analysierte, zurückgegriffen.[876] Birkhofer sammelte dazu aus einem technischen Wörterbuch über tausend Verben und schränkte diese Anzahl auf 222 „relativ eigenständige“ Verben ein.[877]

Bei der *syntaktischen Unterscheidung* von Verben wird im Deutschen zunächst in Vollverben (z.B. *bohren, fallen, stehen*) und Hilfsverben (z.B. *sein, werden*) unterschieden. Vollverben haben eine lexikalische Bedeutung[878] und können allein das Prädikat eines Satzes bilden. Hilfsverben hingegen „helfen“ lediglich anderen Verben, bestimmte Verbformen zu bilden (bspw. bei der Bildung von Zeitformen). Es werden im weiteren Verlauf lediglich Vollverben berücksichtigt. Die (Voll-)Verben werden anschließend in transitive und intransitive Verben unterschieden. Transitive Verben drücken die Wirkung eines Subjekts auf ein Akkusativobjekt im Satz aus.[879] Das Akkusativobjekt wird bei einer Passivierung des Satzes zum Subjekt des Satzes. Demgegenüber kann bei intransitiven Verben kein Akkusativobjekt vorkom-

874) Aufgrund der Verwendung der Formblätter der Karl Schumacher Maschinenbau GmbH, die als Unternehmen vornehmlich in Deutschland tätig wird, und der damit verbundenen Abfassung der Formblätter in deutscher Sprache wird im Folgenden lediglich „in Deutsch gedacht“.

875) Vgl. Birkhofer (1980), S. 69.

876) Vgl. Birkhofer (1980), insbesondere S. 72.

877) Vgl. Birkhofer (1980), S. 70.

878) Die lexikalische Bedeutung eines Verbs entspricht der Bedeutung, die durch ein Wörterbuch verzeichnet wird. Demgegenüber stehen unterschiedliche textuelle Bedeutungen, die ein Verb in verschiedenen Texten einnehmen kann. Die lexikalische Bedeutung entspricht einer Abstraktion von den einzelnen textuellen Bedeutungen (vgl. Weinrich (2003), S. 21).

879) Streng genommen findet die Wirkung nicht im Satz, sondern in der Realität statt.

men.[880] Die transitiven Verben zeigen eine auf ein Objekt gerichtete Handlung und die intransitiven Verben zeigen das Ergebnis oder den Zustand einer Handlung.[881]

Die Menge der von Birkhofer ermittelten transitiven Verben zur Darstellung von Funktionen überwiegt deutlich die Menge der intransitiven Verben, deshalb wird eine zusätzliche Gliederungsebene bei den transitiven Funktionen (`transitive_funktion`) eingeführt. Es wird unter der Ebene der transitiven Funktionen in positive, negative und neutrale Konzepte für Funktionen unterschieden (`positive_funktion`, `negative_funktion`, `neutrale_funktion`). Positive Funktionen führen im Allgemeinen zu erwünschten Handlungen (`positionieren`, `bohren`) und negative Funktionen führen dementsprechend im Allgemeinen zu unerwünschten Handlungen (`verschleissen`, `beanspruchen`). Neutrale Funktionen lassen sich weder einer erwünschten noch einer unerwünschten Handlung zuordnen (`skalieren`, `regeln`).

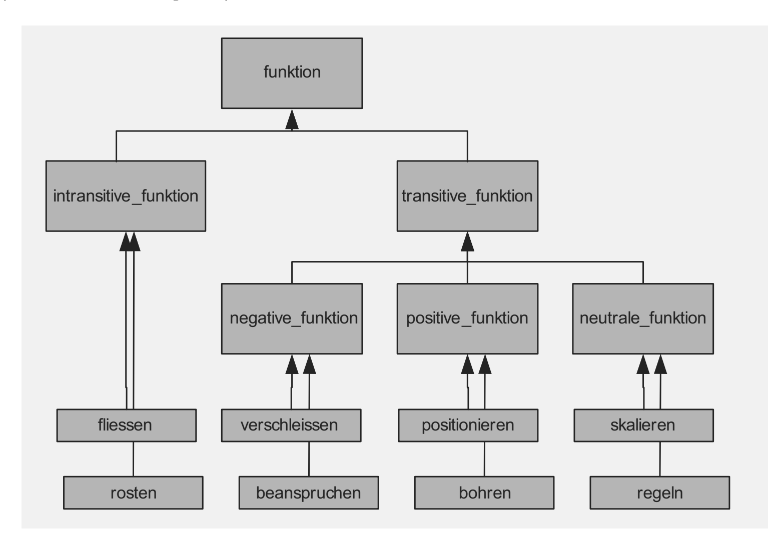


Abbildung 57: Ausschnitt Unterkonzepte von Konzept `funktion`

880) Transitive Verben können jedoch auch intransitiv verwendet werden, wenn transitive Verben ohne Akkusativobjekt einen korrekten Satz bilden. Bei intransitiv verwendeten transitiven Verben wird dabei in der Regel das Akkusativobjekt einfach weggelassen. Es ist dann aus dem Kontext zu erschließen (z. B.: Die Maschine transportiert (den Werkstückträger).).

881) An dieser Stelle eröffnet sich auch noch eine weitere Möglichkeit, Verben zu klassifizieren. Bei einer *semantischen Unterscheidung* möglicher Verben lässt sich im Deutschen eine Einteilung hinsichtlich *Tätigkeitsverben*, *Vorgangsverben* und *Zustandsverben* vornehmen. Dabei drücken Tätigkeitsverben (auch Handlungsverben genannt) aus, dass ein Subjekt handelt (*Die Maschine bohrt.*). Vorgangsverben bezeichnen eine Veränderung, die einem Subjekt widerfährt (*Die Maschine fällt vom Haken.*). Mit Zustandsverben wird etwas sich nicht Veränderndes bezeichnet (*Die Maschine steht in der Halle.*). Aus Übersichtlichkeitsgründen wird auf eine weitere Gliederungsform jedoch verzichtet, weil sie zum einen nicht notwendig für die weitere Argumentationsführung erscheint (eine zusätzliche Ebene wirkt eher kontraproduktiv, weil sie zu einer höheren Komplexität führt). Zum anderen ist gerade die semantische Unterscheidung nicht frei von Doppeldeutigkeiten (so wäre es etwa denkbar, dass *bohren* auch einem Subjekt *Maschine* widerfährt, etwa wenn weitere Bauteile an ihr angebracht werden), so dass eine allgemeingültige Taxonomie nicht ohne komplexe Einschränkungen möglich wird.

Die entwickelte Taxonomie erweitert die FMEA-Ontologie, indem sie das Konzept `funktion` weiter spezifiziert (vgl. Abbildung 57). Die Abbildung verdeutlicht diese Aussage und gibt einige Beispiele für untergeordnete Funktionskonzepte in der FMEA-Ontologie an (z. B. `rosten`, `beanspruchen`, `bohren` und `regeln`).[882] Mögliche Instanzen für Konzepte von Funktionen, die ein Systemelement erfüllt, können mit Hilfe der taxonomischen Struktur *Funktionen* ermittelt werden.[883]

882) Die gesamten Konzepte finden sich im Anhang in der FMEA-Ontologie, S. 336.

883) Siehe hierzu insbesondere das Unterkapitel 8.3.3, S. 260 ff.

7.5 Instanzen-Ebene

7.5.1 Beschreibung der Ebene

Das während der Erstellung der FMEA gewonnene Wissen wird als Instanzen der Konzepte und der Relationen der Ontologie in einer Wissensbasis repräsentiert. Exemplarisch werden im Folgenden Instanzen einiger Konzepte dargestellt.

7.5.2 FMEA-Formular C 5960 A – Bedienpult

Im folgenden Beispiel instanziiert das Objekt `c_5960_a_fmea_bedienpult` das Konzept `fmea`.[884] Systembezeichnung und FMEA-Nummer werden als Zeichenketten mit der Instanz verbunden. Durch den booleschen Wert *true* wird festgelegt, dass es sich um eine System-FMEA Produkt handelt. Andersherum wird durch *false* festgelegt, dass es sich nicht um eine System-FMEA Produkt und damit um eine System-FMEA Prozess handelt. Alle weiteren Relationen verbinden diese Instanz (`c_5960_a_fmea_bedienpult`) mit weiteren Instanzen von Konzepten der Ontologie:

```
c_5960_a_fmea_bedienpult:fmea[
      ist_system_fmea_produkt->"true";
      hat_fmea_nr->"C 5960 A";
      betrachtet_system_von_typ_modell_fertigung_charge->"9+1_Spindel";
      hat_verantwortung->>ksm_mueller;
      wurde_erstellt->_03062000;
      untersucht_systemelement->c_5960_a_bedienpult].
```

Die mit der oben genannten FMEA verbundene Instanz von *Datum* wird, wie folgt, kodiert:

```
_03062000:datum[
                 jahr->2000;
                 monat->6;
                 tag->3].
```

884) Die Bezeichnung einer Instanz des Konzepts `fmea` setzt sich aus der zugehörigen FMEA-Nummer, der Kategorisierung „fmea“ und der Bezeichnung des betrachteten Systemelements zusammen. Die einzelnen Partikel werden mittels Unterstriche zu einem Gesamtausdruck zusammengesetzt. Der Ausdruck „c_5960_a_fmea_bedienpult“ gibt somit an, dass einem FMEA-Formblatt entsprechend das Systemelement „Bedienpult“ mit der FMEA-Nummer „C 5960 A“ untersucht wird.

Um eine existierende FMEA entsprechend einem FMEA-Formblatt als Wissen in der FMEA-Ontologie zu hinterlegen, ist es notwendig, dass untersuchte Systemelement selbst zunächst als Instanz zu hinterlegen. Die Menge der Instanzen, die über eine `ist_teil_von_systemelement`-Relation miteinander verbunden werden, ergibt die untersuchte Systemstruktur einer FMEA. Die Instanz `c_5960_a_bedienpult` des F-Moleküls *Systemelement* wird folgendermaßen in der Wissensbasis repräsentiert:

```
c_5960_a_bedienpult:bedieneinheit[
      hat_systemelementbezeichnung->"Bedienpult";
      ist_teil_von_systemelement->>c_5960_a_neun_eins_spindelschrauber].
```

Der Instanz `c_5960_a_bedienpult` wird folgend exemplarisch eine Funktion als Instanz über die Relation `ist_funktion_von_systemelement` zugeordnet:

```
c_5960_a_umschaltung_automatik_oder_manuellbetrieb:schalten[
      hat_funktionsbezeichnung->"Umschaltung Automatik-/Manuellbetrieb";
      ist_funktion_von_systemelement->>c_5960_a_bedienpult].
```

Auch als Instanz des Konzepts `einrichtung` lässt sich der Ausdruck `c_5960_a_bedienpult` in der Wissensbasis hinterlegen, nachdem er bereits als Instanz des Konzepts `einheit` erfasst wurde:[885]

```
c_5960_a_bedienpult:einrichtung.
```

885) Hiermit lässt sich das Bedienpult als Feld der Tabelle 2, S. 65, – als Feld der Matrix im Schnittpunkt von Betrachtungskriterium und Betrachtungsebene – auffassen. Mit den zwei Relationen zu `einrichtung` und `funktionseinheit` (als Oberkonzept von `bedieneinheit`) entspricht es einer funktionalen Einrichtung. Diese Kodierung dient an dieser Stelle lediglich als Beispiel zur Verdeutlichung, sie findet sich nicht im Anhang.

8 Prototyp OntoFMEA

8.1 Aufbau von OntoFMEA

In diesem und den folgenden Unterkapiteln von Kapitel 8 wird der Prototyp eines Wissensbasierten Systems auf der Basis von Ontologien und FMEAs vorgestellt. Als Herzstück verwendet der Prototyp, der verkürzt als „OntoFMEA“ bezeichnet wird, die in den vorigen Kapiteln entwickelte FMEA-Ontologie mit ihrer zugehörigen Wissensbasis. Insbesondere die Funktionalität des Prototyps dient später zur Argumentation bei der eingehenderen Betrachtung der wissenschaftlichen Problemstellung, die mit dieser Arbeit ein Stück weit gelöst werden soll.

OntoFMEA ist ein vornehmlich in Java geschriebenes, webbasiertes Softwaresystem. Die Verwendung erfordert die vorherige Installation

- eines Java Development Kits von Sun[886],
- des Jakarta-Tomcat JSP-Servers der Apache Software Foundation[887],
- der Ontobroker Inferenzmaschine der Ontoprise GmbH[888] und
- eines aktuellen Webbrowsers[889].

Der konzeptionelle Aufbau von OntoFMEA entspricht der skizzierten Darstellung eines Wissensbasierten Systems der Abbildung 21 von S. 93.[890]

8.2 Starten von OntoFMEA

8.2.1 Vorbereitung des Starts von OntoFMEA

Um OntoFMEA zu starten, muss zuerst die Inferenzmaschine Ontobroker mit der zum Einsatz gelangenden erweiterten FMEA-Ontologie (inkl. Wissensbasis) gestartet werden. Weil oftmals die verwendeten Ontologien so umfangreich sind, dass Ontobroker in der hier verwendeten Standardkonfiguration „abstürzt“, erfolgt die Ausführung eines Startskripts, das die spezifische Installation berücksichtigt. Anschließend erfolgt der Start von Tomcat. Sind beide

886) Hier wird die Java 2 Plattform, Standard Edition, in der Version 1.4.2 von http://java.sun.com eingesetzt (Zugriff am 19.10.2004).

887) Hier wird der Server in der Version 4.1.31 von http://jakarta.apache.org eingesetzt (Zugriff am 19.10.2004).

888) Hier wird die Inferenzmaschine in der Version 3.7 von http://www.ontoprise.de eingesetzt (Zugriff am 19.10.2004).

889) Hier wird der Opensource-Webbrowser Firefox des Mozilla-Projekts in der Version 0.9.3de von http://www.firefox-browser.de eingesetzt (Zugriff am 19.10.2004). Die Verwendung dieses Webbrowsers ermöglicht im Gegensatz zu Versionen des weit verbreiteten Internet-Explorers der Microsoft AG die korrekte Darstellung von Stylesheets. Daher sollte der Webbrowser Firefox verwendet werden, wenn alle grafischen Funktionen von OntoFMEA berücksichtigt werden sollen.

890) Siehe zum weiteren Verständnis von Ontobroker die Ausführungen im dazugehörigen Kapitel 4.1.4.3.2, S. 91 ff.

Softwaresysteme ordnungsgemäß hochgefahren, so kann mittels des Webbrowsers die Startseite von OntoFMEA aufgerufen werden (vgl. hierzu auch die Abbildung 58, Abbildung 59 und Abbildung 60).[891]

```
C:\WINNT\system32\cmd.exe
loading external builtin: IsOdd
OntoBroker V 3.7 on port 1234 launching ...
launching -p 1234 -flo C:/Dokumente und Einstellungen/lars.dittmann/Desktop/fmea
.flo
explanation off
cache off
frontside cache off
internal db off
materializeaxioms off
automaterialize off
deep namespaces
loading C:/Dokumente und Einstellungen/lars.dittmann/Desktop/fmea.flo
InferenzServer up and running ...
_
```

Abbildung 58: Ausführung der Startroutine für Ontobroker

```
Start Tomcat
22.01.2006 15:38:33 org.apache.coyote.http11.Http11Protocol init
INFO: Initializing Coyote HTTP/1.1 on port 8080
Starting service Tomcat-Standalone
Apache Tomcat/4.1.30
22.01.2006 15:38:38 org.apache.struts.util.PropertyMessageResources <init>
INFO: Initializing, config='org.apache.struts.util.LocalStrings', returnNull=tru
e
22.01.2006 15:38:38 org.apache.struts.util.PropertyMessageResources <init>
INFO: Initializing, config='org.apache.struts.action.ActionResources', returnNul
l=true
22.01.2006 15:38:40 org.apache.struts.util.PropertyMessageResources <init>
INFO: Initializing, config='org.apache.webapp.admin.ApplicationResources', retur
nNull=true
22.01.2006 15:38:49 org.apache.coyote.http11.Http11Protocol start
INFO: Starting Coyote HTTP/1.1 on port 8080
22.01.2006 15:38:49 org.apache.jk.common.ChannelSocket init
INFO: JK2: ajp13 listening on /0.0.0.0:8009
22.01.2006 15:38:49 org.apache.jk.server.JkMain start
INFO: Jk running ID=0 time=50/130  config=C:\Programme\Apache Group\Tomcat 4.1\c
onf\jk2.properties
_
```

Abbildung 59: Ausführung der Startroutine für Tomcat

891) OntoFMEA wird dabei unter der Adresse http://localhost:8080/onto im Browser aufgerufen.

8.2.2 Startseite von OntoFMEA

Die Abbildung 60 zeigt die Startseite des Prototyps von OntoFMEA, wie sie über den Webbrowser zu erreichen ist. Auf der linken Seite findet sich ein Menü für die weitere Navigation zu den Funktionen, die vom Prototyp erfüllt werden, und im Zentrum werden bereits durchgeführte FMEAs, über die das System verfügt, präsentiert. Auf der obersten Ebene wird beim Prototyp OntoFMEA zwischen Gesamtanwendungen und Einzelanwendungen unterschieden. Die Gesamtanwendungen ermöglichen den Einsatz des Systems als Ganzes bspw. in einem Unternehmen. Die Einzelanwendungen dienen in erster Linie dazu, punktuelle Eingriffe in die Wissensbasis oder Auswertungen aus der Wissensbasis auch ohne einen anwendungsspezifischen Gesamthintergrund (wie bspw. die spezifische Durchführung einer FMEA) zu ermöglichen.

Der Einzelanwendungsbereich lässt sich hinsichtlich des Funktionsumfangs zweiteilen. Zum einen findet sich der Bereich, der die Veränderung der Wissensbasis im engeren Sinne umfasst. Zum anderen gibt es den Bereich, der das Einsehen/Analysieren der Wissensbasis im engeren Sinne erlaubt. Im ersten Bereich werden einzelne Instanzen manipuliert, d. h., es geht um die Anwendung von Ausschnitten aus der Wissensbasis (Funktion *Einzelne Instanzen*, Abbildung 60), und im zweiten Bereich geht es um die Anwendung der gesamten Wissensbasis (Funktion *Analyse*, ebenfalls Abbildung 60).[892]

Nähere Erläuterungen zum Einzelanwendungsbereich von OntoFMEA sind Dittmann, Rademacher (2004) zu entnehmen. Um den Umfang an dieser Stelle zu begrenzen, wird im Folgenden lediglich kurz auf die berücksichtigten Gesamtanwendungen als Hauptanwendungsfälle eingegangen. Dem Leser soll so das Potenzial von OntoFMEA verdeutlicht werden. In diesem Zusammenhang wird jeweils kurz auf modellierungstechnische Besonderheiten der Anwendung eingegangen (besonders auf eine Reihe von Regeln).

892) Bei genauerer Betrachtung der Abbildung 60 findet sich zusätzlich noch der Menüpunkt „Sonderfunktionen“. Dieser Menüpunkt wird intern genutzt, um Konzepte innerhalb der Ontologie zu sortieren, damit eine einfachere Lesbarkeit ermöglicht wird. Diese Anwendung dient nicht unmittelbar einer Nutzensteigerung während des „laufenden Betriebs“, wie sie mit diesem System nachgewiesen werden soll, weil sie speziell zu Anfang Verwendung findet, wenn zahlreiche neue Konzepte in die Ontologie aufgenommen werden, und wird deshalb nicht weiter ausgeführt.

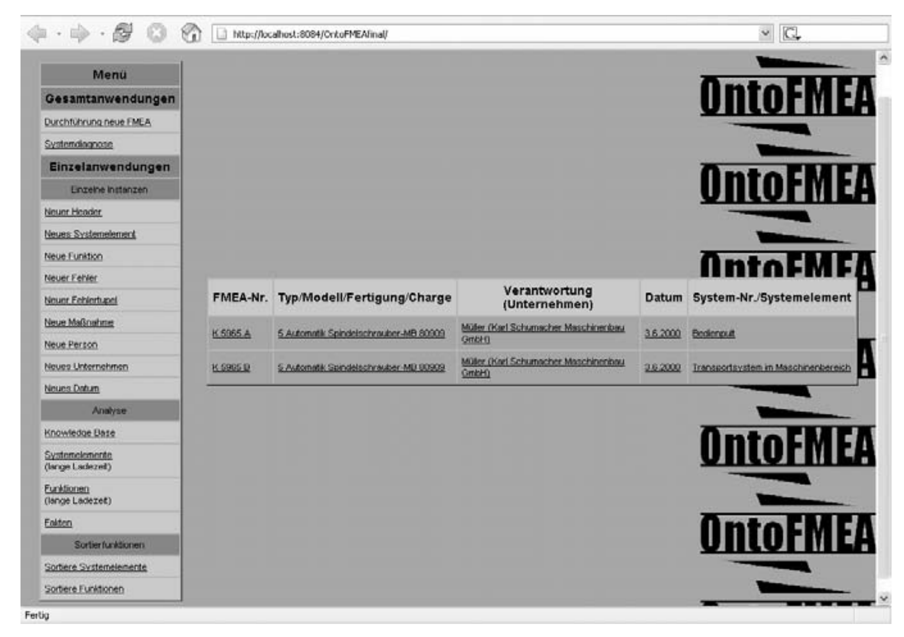


Abbildung 60: Bildschirmabzug *Startseite OntoFMEA*

Derzeit unterstützt OntoFMEA die folgenden Gesamtanwendungen:[893]

- **Durchführung neue FMEA**
  Dieser Menüpunkt beherbergt das Herzstück des Softwaresystems. Er ermöglicht die Erstellung einer FMEA. Der Erstellungsprozess wird durch OntoFMEA unterstützt, indem Vorschläge für mögliche Subsystemelemente, mögliche Funktionen und mögliche Fehler unterbreitet werden.[894] Das Vorgehen entspricht hierbei exakt dem Vorgehen gemäß VDA-Band 4 (2003).

- **Anzeigen erstellter FMEAs**
  Die mittig auf dem Startbildschirm angezeigten, bereits angelegten FMEAs können mittels Auswahl eingesehen werden. Der Anwender kann hier intuitiv und „frei“ durch das betrachtete System, seine zugeordneten Funktionen, Fehler, Fehlertupel und Maßnahmen browsen. Alle Informationen einer FMEA gemäß VDA sind dabei abrufbar.

- **Systemdiagnose**
  Dieser Menüpunkt dient als erster Nachweis der sinnvollen Weiterverwendung des bereitgehaltenen Wissens einer FMEA. Die Funktion stellt vorläufig eine beispielhafte, prototypische Anwendung dar, die noch nicht in Gänze ausgereift ist. Der Menüpunkt

893) Im Folgenden werden diese Gesamtanwendungen linguistisch verkürzt als Menüpunkte zusammengefasst, weil sich jeweils hinter dem bezeichnenden Menüpunkt die dazugehörige Gesamtanwendung verbirgt.

894) Vgl. hierzu auch das Kapitel 8.3, S. 256 ff.

erlaubt die Ermittlung von Ursachen für aufgetretene Fehler, d. h., es kann gezielt nach einem Fehler gesucht werden. Anschließend ermittelt das System mögliche Ursachen für den Fehler, so dass der Anwender mit diesem Wissen eine Fehlerbehebung angehen kann. Ein möglicher Einsatzort für ein solches System liegt bspw. in einer Werkstatt, die Reparaturen an defekten Systemen, für die bereits FMEAs erstellt und in OntoFMEA repräsentiert wurden, vornimmt.

8.3 Menüpunkt Durchführung neue FMEA

8.3.1 Stammdaten anlegen

Nachdem der Menüpunkt *Durchführung neue FMEA* angewählt wurde, erhält der Benutzer eine Ansicht, die der Abbildung 61 entspricht. Auf der linken Seite findet sich während der gesamten Durchführung einer FMEA das Phasenmodell der fünf Schritte zur Erstellung einer System-FMEA gemäß VDA.[895] Dabei wird im weiteren Verlauf der Anwendung der jeweils aktuelle Schritt durch Unterstreichung und Kursivsetzung hervorgehoben.

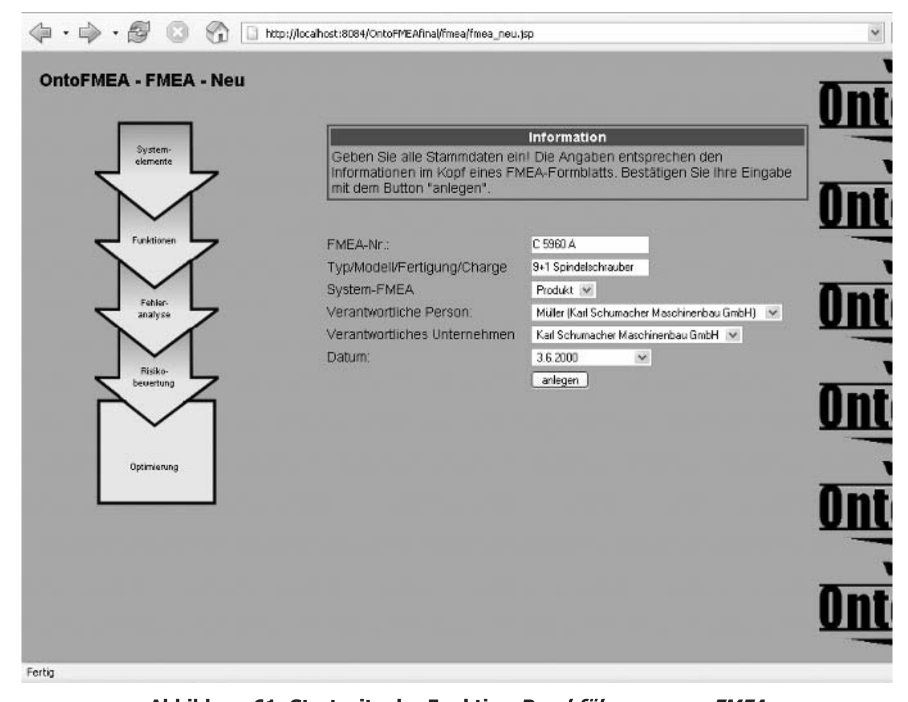


Abbildung 61: Startseite der Funktion *Durchführung neue FMEA*

895) Vgl. VDA-Band 4 (2003), S. 15. Siehe hierzu auch die Ausführungen in Kapitel 3.1.4.1, S. 46, insbesondere Abbildung 9, S. 47.

Zu Beginn der Durchführung müssen die Stammdaten, die sich im Kopf (Header) eines FMEA-Formblatts finden, angegeben werden. Aus diesen Informationen soll das betrachtete System unmittelbar (eindeutig) erkennbar werden.[896] Hierzu zählen die Angaben einer FMEA-Nummer, von Typ/Modell/Fertigung/Charge, die Art der System-FMEA (Produkt/ Prozess), der verantwortlichen Person, des verantwortlichen Unternehmens und des Datums der FMEA-Erstellung.

Falls es möglich ist, unterstützt OntoFMEA die Durchführung einer FMEA durch zwei rot umrandete Hinweisfelder mit den Überschriften „Ergebnis“ und „Information“.[897] Während das Hinweisfeld „Ergebnis“ sich immer auf die zuletzt durchgeführte Aktion bezieht und Auskunft über den Erfolg oder Mißerfolg gibt, liefert das Hinweisfeld „Information“ Auskünfte über die nächste anstehende Aktion.

Angelegte Informationen werden im System direkt der Ontologie als „Fakten“ (instanziierte F-Atome/F-Moleküle) hinzugefügt und stehen somit unmittelbar als Wissen zur Verfügung.[898]

8.3.2 Systemelemente und Systemstruktur neu anlegen

Nach Eingabe der Stammdaten erfolgt die eigentliche Analyse über das Anlegen von Systemelementen des Betrachtungsobjekts.[899] Zunächst wird das Gesamtsystem als Instanz generiert und anschließend werden die untergeordneten Systemelemente angelegt. Beginnend mit einem Oberkonzept (hier: `schraubmaschine`) wird die Systemstruktur top-down für das betrachtete System angelegt. OntoFMEA macht hierzu Vorschläge für mögliche Systemelemente, indem es seine Wissensbasis nach verwandten Systemelementen durchsucht und diese als Vorschläge äußert.[900] Die Abbildung 62 zeigt dieses Vorgehen. Es wurde bereits eine Instanz von `schraubmaschine` mit der Bezeichnung „`Neun Eins Spindelschrauber`“ angelegt.[901] Der Maschine `Neun Eins Spindelschrauber` wurde das Systemelement `Bedienpult` als Instanz (intern: `c_5960_a_bedienpult`) des Konzepts `bedienein-`

896) Vgl. VDA-Band 4 (2003), S. 27.

897) Die Bedingung der Möglichkeit ergibt sich unmittelbar aus den folgenden Aussagen des Abschnitts. So kann das Hinweisfeld „Ergebnis“ nur sinnvoll erscheinen, wenn zuvor eine Aktion, die ein Ergebnis erzielt, durchgeführt wurde.

898) Im weiteren Verlauf der beispielhaften Erläuterungen zur Durchführung einer FMEA wird auf das Formblatt mit der FMEA-Nr. C 5960 A (9+1 Spindelschrauber, Bedienpult) der Karl Schumacher GmbH, das sich im Anhang dieser Arbeit auf S. 332 findet, zurückgegriffen.

899) Zum Konstrukt der Systemelemente siehe auch Kapitel 7.4.2, S. 245 ff.

900) Mit der Generierung von Vorschlägen kommt es somit durch OntoFMEA zu einer Wissenswiederverwendung. Bereits enthaltenes Wissen aus der Wissensbasis im weiteren Sinne wird dem Benutzer zur Arbeitserleichterung zur Verfügung gestellt.

901) Um den Benutzer nicht zu verwirren, wird von OntoFMEA lediglich die über die Relation `hat_-systemelementbezeichnung` zugehörige Systemelementbezeichnung „`Neun Eins Spindelschrauber`“ dargestellt.
Die eigentliche Instanz `c_5960_a_neun_eins_spindel-schrauber` wird lediglich intern, d.h. für den Benutzer nicht unmittelbar sichtbar, verwendet. Um die Unterscheidung im weiteren Textverlauf zu verdeutlichen, werden zugeordnete Bezeichnungen zu Anfang groß geschrieben und nicht mittels Unterstrichen durchgekoppelt.

heit aus der FMEA-Ontologie hinzugefügt.[902] Erkennbar wird die bereits angelegte Systemstruktur an der tabellarischen Einrückung der jeweils untergordneten Systemelemente im Gegensatz zum jeweiligen übergeordneten Systemelement.

OntoFMEA macht auf der Basis bereits angelegter FMEAs Vorschläge für weitere Systemelemente und deren Eingruppierung in der Systemstruktur. So ermittelt im vorliegenden Bildschirmabzug OntoFMEA neben dem Konzept bedieneinheit bspw. noch die weiteren Konzepte gehaeuse-, hydraulik-, pneumatik-, prozessdatenerfassungs- und transporteinheit als mögliche Systemelemente einer schraubeinheit. Die Konzepte elektrischer_schalter und schluesselschalter werden als mögliche untergeordnete Systemelemente einer bedieneinheit vorgeschlagen.

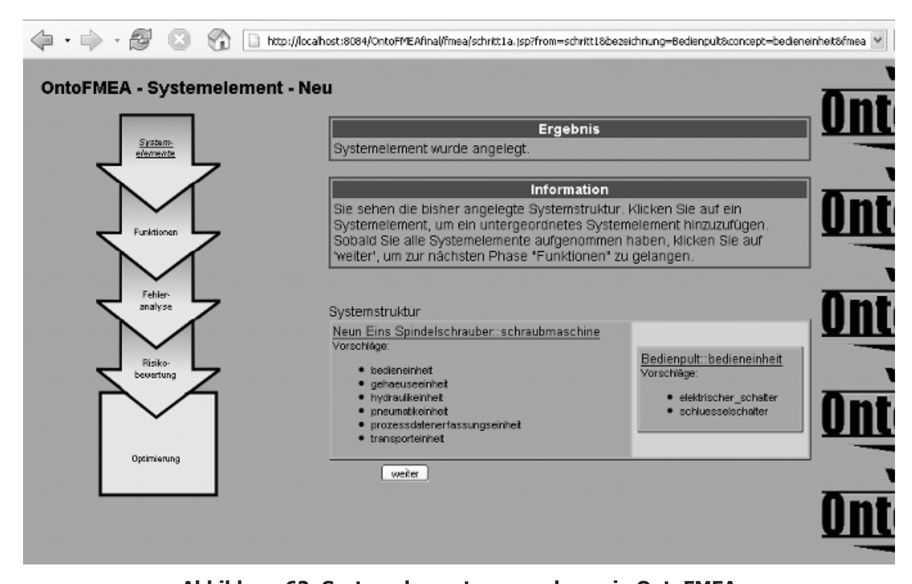


Abbildung 62: Systemelemente neu anlegen in OntoFMEA

Um ein Systemelement anzulegen, muss der Anwender lediglich das jeweilige übergeordnete Systemelement, dem das anzulegende Systemelement untergeordnet werden soll, durch Anklicken am Bildschirm anwählen. Anschließend gibt der Benutzer die Bezeichnung für das untergeordnete Systemelement (bspw. den gemachten Vorschlag) und das zugehörige Konzept aus der FMEA-Ontologie an. OntoFMEA legt anschließend dieses Fakt (als instanziiertes F-Molekül) in der Wissensbasis ab. Zu diesem Zweck wird intern eine Instanz angelegt, bei der automatisch die Bezeichnung klein geschrieben und mit der FMEA-Nummer durchgekoppelt wird.[903]

902) Das erfolgreiche Anlegen wird im Hinweisfeld „Ergebnis“ dabei bestätigt.

903) Siehe hierzu auch die Ausführungen in Fn. 901, S. 257.

Bei der Ermittlung von Vorschlägen nutzt OntoFMEA in der Hauptsache die folgende Regel:

```
FORALL V,W,X,Y,Z  W:systemelement[hat_moegliches_unterkonzept=>>V] <-
            directisa_(W,X) AND
            directisa_(Y,X) AND
            Y[ist_zusammengesetzt_aus_systemelement->>Z] AND
            directisa_(Z,V).
```

Die Relation `hat_moegliches_unterkonzept` verknüpft mögliche Unterkonzepte mit einer Instanz des Konzepts `systemelement`. Als Vorschläge können zugehörige Instanzen und Konzepte mit dem Built-In `directisa_(Y,X)` im Rumpfteil der Regel ermittelt werden. Mit Hilfe des Built-Ins werden zu einem Konzept X die in der Wissensbasis zugeordneten Instanzen Y mit der Inferenzmaschine Ontobroker ermittelt.[904] Das Built-In entspricht der folgenden Regel in F-Logic:[905]

```
FORALL Y,X  Y[is_direct_instance->>X] <-
            Y:X AND NOT (EXISTS Z((Y:Z AND Z::X))).
```

Ein mögliches Systemelement, das beim Anlegen der Systemstruktur berücksichtigt werden sollte und deshalb von OntoFMEA vorgeschlagen wird, lässt sich beispielhaft für die „Vorschlags"-Regel folgendermaßen kommentieren:

Vorraussetzung für einen Schlussfolgerung sind, dass in der Wissensbasis bereits eine Instanz Z (`k_5965_a_bedienpult`) zu einem Konzept V (`bedieneinheit`) existiert.[906] Diese Instanz Z (`k_5965_a_bedienpult`) ist einer Instanz Y (`k_5965_fuenf_automatik_spindelschrauber`) untergeordnet, d.h. die Instanz Y ist aus der Instanz Z (u. a.) zusammengesetzt. Ferner sind einem Konzept X (`schraubmaschine`) die Instanzen Y (`k_5965_fuenf_automatik_spindelschrauber`) und W (`c_5960_neun_eins_spindelschrauber)` zugeordnet.

904) Es handelt sich somit nicht um die Ermittlung von Unterkonzeptbeziehungen aus der konzeptuellen Modellierung, die oftmals ebenfalls mit dem Kürzel „is a" dargestellt werden.

905) Siehe hierzu auch die Ausführungen in Kapitel 7.2.3.5, S. 226, hier insbesondere die Fn. 837 und 838, S. 228.

906) In dem Beispiel wird die Funktion der Regel erläutert, indem in Klammern Instanzen und Konzepte exemplarisch genannt werden, wie sie auch in der FMEA-Ontologie im Anhang zu finden sind. Wie bereits erwähnt wurde, werden die Instanzen mit der FMEA-Nummer durchgekoppelt und Konzepte als ein Wort dargestellt.

Aus diesen Bedingungen lässt sich plausibel schließen, dass der Instanz W (c_5960_neun_eins_spindelschrauber) auch eine *mögliche Instanz* des Unterkonzepts V (bedieneinheit) als untergeordnetes Systemelement zugeordnet werden kann.[907] Somit lässt sich das Konzept V (bedieneinheit) als möglicher Vorschlag von OntoFMEA für den Aufbau einer Systemstruktur nutzen.

8.3.3 Funktionen und Funktionsstruktur neu anlegen

Nach dem erfolgreichen Anlegen der Systemstruktur gelangt der Anwender zum 2. Schritt der FMEA-Durchführung: Den einzelnen Systemelementen werden Funktionen zugeordnet. Wie aus der Abbildung 63 deutlich wird, macht das System – sofern möglich – erneut Vorschläge bezüglich möglicher Konzepte für mögliche Funktionen, die von einem Systemelement unterstützt werden sollen. Hauptsächlich wird zur Vorschlagsgenerierung auf die folgende Regel in der Ontologie zurückgegriffen:[908]

```
FORALL U,V,W,X,Y,Z V:systemelement
      [hat_moegliches_funktionskonzept=>>U;
       hat_moegliche_funktionsbezeichnung->Z] <-
                directisa_(V,W) AND
                directisa_(X,W) AND
                X[hat_funktion->>Y] AND
                Y[hat_funktionsbezeichnung->Z] AND
                directisa_(Y,U).
```

Mit Hilfe der Regel verknüpft die Relation hat_moegliches_funktionskonzept mögliche Funktionen mit einem systemelement. Als Vorschläge, die von OntoFMEA angeboten werden, können zugehörige Instanzen zu den vorzuschlagenden Konzepten unter Verwendung des Aufrufs des Built-Ins directisa_(X,Y) ermittelt werden. Die Relation hat_moegliche_funktionsbezeichnung verknüpft zur weiteren Unterstützung des Benutzers mögliche Funktionsbezeichnungen mit einer Instanz von systemelement. Die Instanzen für mögliche Funktionsbezeichnungen werden ebenfalls von OntoFMEA angeboten (auf dem Bildschirmabzug der Abbildung 63 bspw. als Verschraubt Getriebe zu erkennen).

907) In der Schlussfolgerung steckt die Annahme, dass Systemelemente, die demselben Systemelement untergeordnet werden, auch bei neu anzulegenden Systemstrukturen, wie sie bei der Durchführung einer FMEA erstellt werden müssen, wiederholt auftreten. Wird bspw. ein Fahrzeug in seine Systemelemente strukturiert, dann werden in der Regel erneut Motor und Getriebe Betrachtungsgegen-stände sein, insbesondere dann, wenn diese in vorangegangen FMEA-Durchführungen schon als Betrachtungsgegenstände Berücksichtigung fanden.
Es handelt sich bei der Regel um eine nicht deduktive, d. h. nicht stringent wahrheitserhaltende, Schlussfolgerung. Weil jedoch diese Schlussfolgerungen, die später auch für die Ermittlung möglicher Funktionen und Fehler genutzt werden, lediglich als Vorschläge für den Benutzer weitere Verwendung finden, bedarf es keiner weiteren Sicherstellung der Integrität der Wissensbasis.

908) Auf eine beispielhafte Erläuterung der Regel wird hier verzichtet, weil sie in ihrer Struktur stark der zuvor erläuterten Regel entspricht. Es wird lediglich zusätzlich zur exemplarisch erläuterten Regel die Funktionsbezeichnung als Instanz ermittelt und dem Benutzer als Option zur Verfügung gestellt.

Durch die Auswahl mittels Anklicken eines Systemelements kann diesem Systemelement eine Funktion zugeordnet werden. Einem Systemelement können beliebig viele Funktionen zugeordnet werden.Die Abbildung 63 zeigt, dass bisher noch keine Funktionen zugeordnet worden sind.

Im Anschluss an die Zuordnung von Funktionen zu Systemelementen erfolgt die Generierung der Funktionsstruktur. Hierbei kann einer Funktion eine weitere Funktion als Teilfunktion über die Relation `ist_teil_von_funktion` zugeordnet werden. Mit der Erstellung der Funktionsstruktur endet der 2. Schritt der Durchführung einer FMEA.

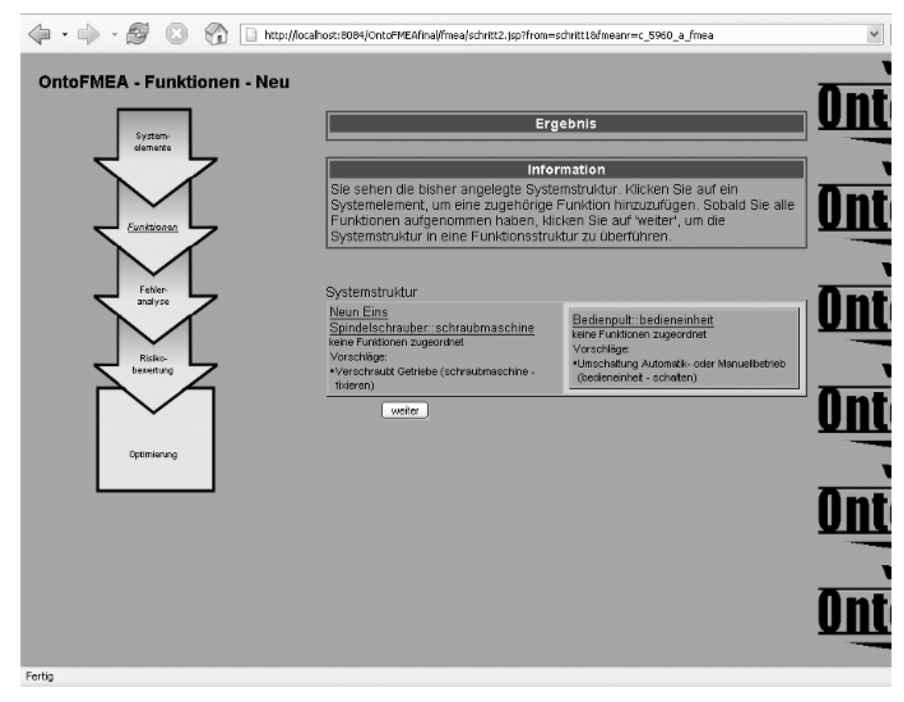


Abbildung 63: Funktionen neu anlegen in OntoFMEA

8.3.4 Fehleranalyse durchführen

Für jedes generierte Systemelement kann in OntoFMEA eine Fehleranalyse durchgeführt werden.[909]

Ähnlich der Zuordnung von Funktionen zu Systemelementen können zu jeder Funktion Fehlermöglichkeiten angelegt werden. Wiederum macht OntoFMEA Vorschläge zu möglichen Fehlern. Anschließend können diese Fehlermöglichkeiten in eine Fehlerstruktur gebracht werden.

909) Vgl. VDA-Band 4 (2003), S. 18.

Die Abbildung 64 verdeutlicht das Vorschlagen von Fehlermöglichkeiten in OntoFMEA. Im Grunde lassen sich zwei Varianten von Vorschlägen unterscheiden. Zum einen wird der Vorschlag gemacht, die eigentliche Funktion lediglich zu negieren. Dieser Vorschlag wird in der Anwendung jeweils fett hervorgehoben (z. B.: `Verschraubt Getriebe nicht oder unzureichend`). Zum anderen wird in der Wissensbasis nach Fehlermöglichkeiten gesucht, die bereits in ähnlichen FMEAs zur betreffenden Funktion Verwendung fanden (z. B.: `Maschinenstoerung` und `Produktionsausfall`).

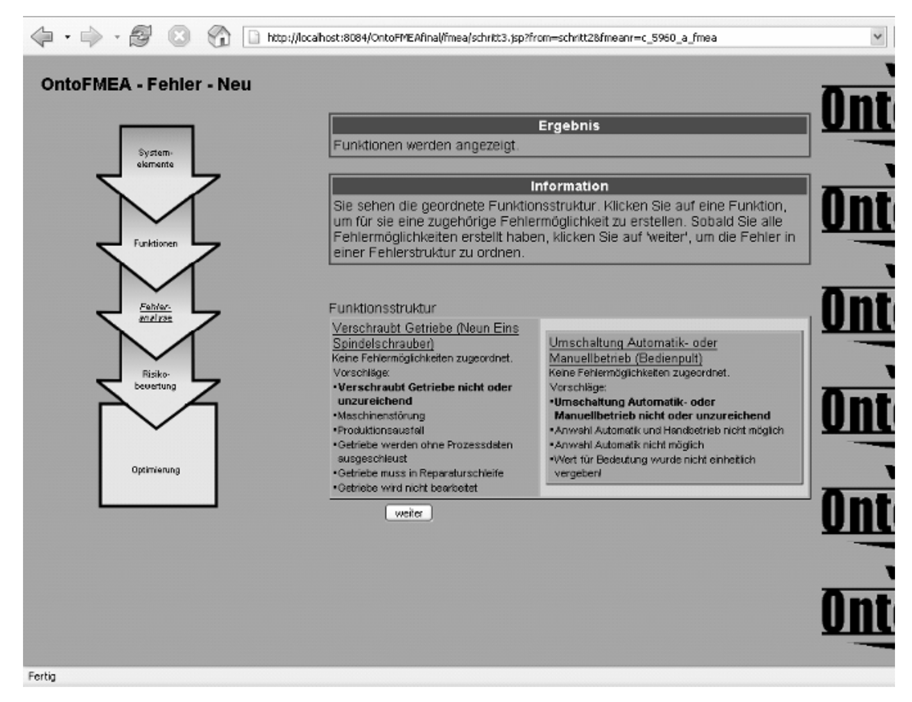


Abbildung 64: Fehleranalyse in OntoFMEA durchführen

Die Suche nach Vorschlägen zu möglichen Fehlern erfolgt hauptsächlich mit Hilfe der folgenden Regel:

```
FORALL V,W,X,Y,Z  W:funktion[hat_moeglichen_fehler=>>V] <-
                  W[ist_funktion_von_systemelement->>Z] AND
                  directisa_(W,X) AND
                  V[verhindert_funktionskonzept=>>X] AND
                  directisa_(Z,Y) AND
                  V[stoert_systemelementkonzept=>>Y].
```

Dabei verknüpft die Relation `hat_moeglichen_fehler` eine Funktion mit möglichen Fehlern. Um diese Regel anwenden zu können, werden noch zwei weitere Regeln benötigt, die die Relationen `verhindert_funktionskonzept` und `stoert_systemele-`

mentkonzept logisch in der Ontologie verknüpfen. Die hiermit ermittelbaren Konzepte werden zur Kategorisierung eines Fehlers verwendet: die Kategorie eines Fehlers ergibt sich aus der Kombination des Konzepts der Funktion, die er verhindert, mit dem Konzept des Systemelements, das durch den Fehler gestört wird:

```
FORALL X,Y,Z      Y:fehler[verhindert_funktionskonzept=>>X] <-
                  Y[verhindert_funktion->>Z] AND
                  directisa_(Z,X).
```

```
FORALL W,X,Y,Z    X:fehler[stoert_systemelementkonzept=>>W] <-
                  X[verhindert_funktion->>Y] AND
                  Y[ist_funktion_von_systemelement->>Z] AND
                  directisa_(Z,W).
```

Bei der ersten Regel verknüpft die Relation verhindert_funktionskonzept Fehler mit dem Konzept einer Funktion, dessen Instanzen von dem Fehler verhindert werden. Bei der zweiten Regel verknüpft die Relation stoert_systemelementkonzept Fehler mit einem Konzept eines Systemelements, dessen Instanzen von dem Fehler gestört werden. Nachdem alle Fehler den Funktionen zugeordnet worden sind, müssen diese in eine Fehlfunktionsstruktur gebracht werden. Erneut erfolgt die Zuordnung einer möglichen Fehlerfolge zu einem möglichen Fehler über die Auswahl des betreffenden möglichen Fehlers. Im Verlauf der Durchführung einer FMEA ergibt sich hierbei in OntoFMEA eine Darstellung gemäß der Abbildung 65. So führen die Fehlerursachen Druckschalter Pneumatik meldet nicht, Pneumatikversorgung fehlt und Schutztuerschalter defekt zum möglichen Fehler Anwahl Automatik und Handbetrieb nicht moeglich und dieser führt zur möglichen Fehlerfolge Maschinenstoerung. Mit Erstellung der Fehlfunktionsstruktur ist der 3. Schritt zur Durchführung einer FMEA erfolgreich abgeschlossen.

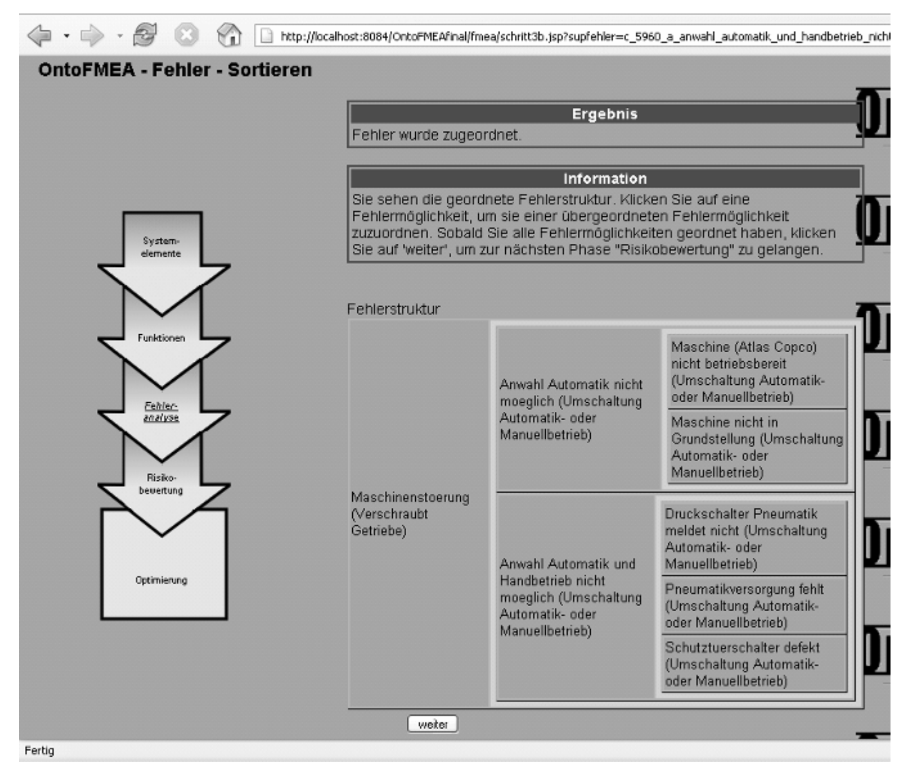


Abbildung 65: Fehlfunktionsstruktur anlegen in OntoFMEA

Zu jedem Fehler innerhalb der Fehlfunktionsstruktur, der eine Ursache für einen weiteren Fehler darstellt und der gleichzeitig seinerseits eine Fehlerfolge besitzt, können im nächsten Schritt „Risikobewertung“ Maßnahmen angegeben und Risikobewertungen durchgeführt werden.

8.3.5 Risikobewertung durchführen

Zuerst werden den Fehlermöglichkeiten Vermeidungs- und Entdeckungsmaßnahmen zugeordnet.

In der Abbildung 66 lässt sich erkennen, dass den Fehlermöglichkeiten bereits Entdeckungsmaßnahmen zugeordnet worden sind. Bisher wurden jedoch noch keine Vermeidungsmaßnahmen explizit zugeordnet.[910]

910) Aufgrund fehlender zeitlicher Ressourcen verfügt der Prototyp von OntoFMEA zum Zeitpunkt der Erstellung dieser Arbeit nicht über die Möglichkeit, Vorschläge bezüglich geeigneter Maßnahmen zu generieren. Aufgrund der vorangegangenen Ausführungen wird jedoch deutlich, dass diese Funktion ohne weiteres entwickelt werden kann, weil sie den anderen Arten von Vorschlägen ähnelt.

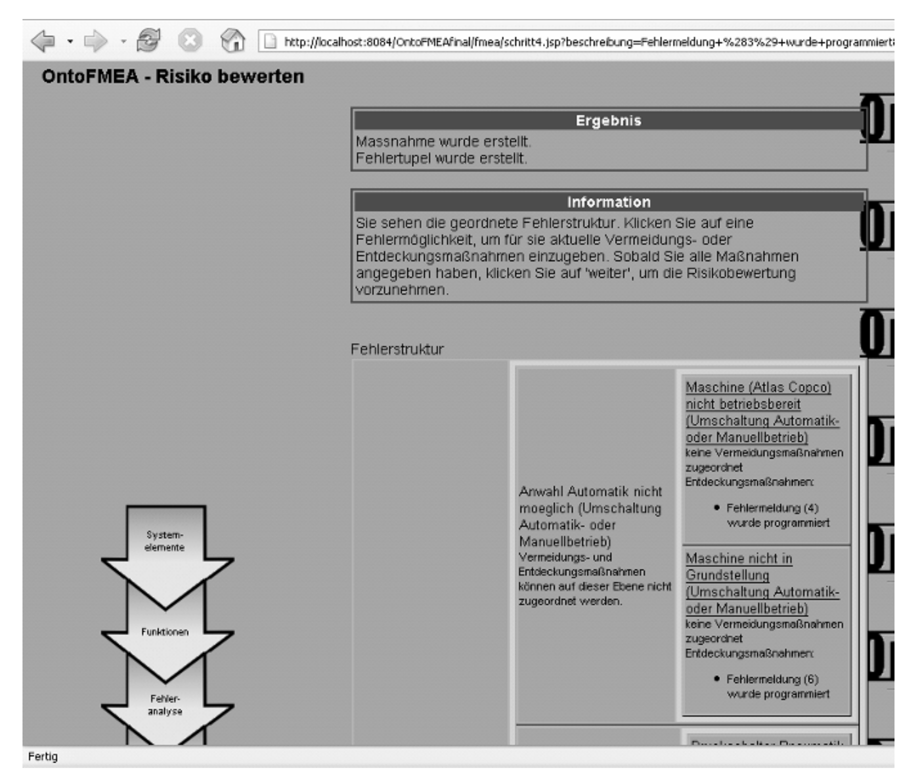


Abbildung 66: Zuordnung von Maßnahmen in OntoFMEA

Wenn eine Maßnahme der Wissensbasis hinzugefügt werden soll, so erstellt OntoFMEA auf der Basis der bisher angelegten Informationen ein Fehlertupel, das auf die Fehlermöglichkeit referenziert.[911]

Anschließend kann für das jeweilige Fehlertupel die eigentliche Risikobewertung vorgenommen werden. Die Bedeutung (B) für eine Fehlerfolge wird hierzu festgelegt sowie die Auftrittswahrscheinlichkeit (A) und die Entdeckungswahrscheinlichkeit (E) für mögliche Fehlerursachen. Die zur Auswahl vorgegebene Werteskala von 1 bis 10 entspricht dabei den Beispielen für Bewertungstabellen des VDA.[912]

Die Abbildung 67 verdeutlicht dieses Vorgehen.

911) Wie bereits zu Ende des vorhergehenden Kapitels erwähnt wurde, lässt sich eine Risikoprioritätszahl nur dann sinnvoll zuweisen, wenn ein Tupel Fehlerursache, Fehler, Fehlerfolge angelegt wurde. Deshalb wird in OntoFMEA der Hinweis gegeben, dass gegebenenfalls Maßnahmen auf einer Ebene nicht zugeordnet werden können, sofern nicht auf die ursächliche Fehlermöglichkeit Bezug genommen werden kann (siehe Abbildung 66).

912) Siehe hierzu auch die Ausführungen in den Kapiteln 3.1.4.1.4, S. 50 f., und 7.3.3.5, S. 237 ff.

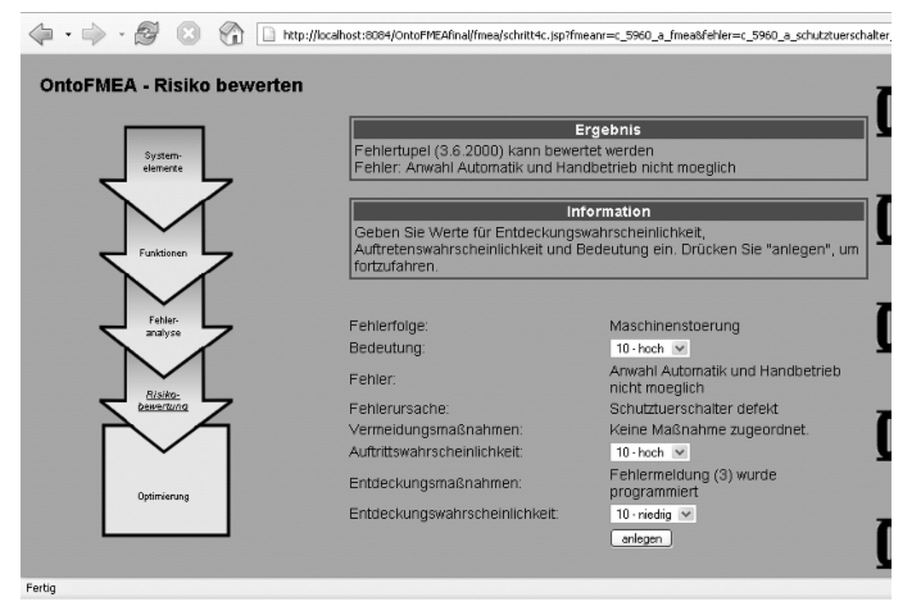


Abbildung 67: Risikobewertung durchführen in OntoFMEA

8.3.6 Optimierung durchführen

Die Risikoprioritätszahl und die zu ihr zusammengesetzten Einzelbewertungen (B, A und E) verdeutlichen die Systemrisiken. Für hohe Risikoprioritätszahlen oder Einzelbewertungen sind weitere Verbesserungsmaßnahmen zu entwickeln.[913]

Einem bereits einer Bewertung unterzogenen Fehler können weitere Fehlertupel zugeordnet werden, die Maßnahmen enthalten, die die Fehlererkennung oder Fehlervermeidung positiv beeinflussen.

Die bereits festgelegten Fehlertupel werden in OntoFMEA gemäß ihrer Wichtigkeit farbig hervorgehoben. Niedrige Risikoprioritätszahlen, für die lediglich wenig Handlungsbedarf besteht, werden im System gelb hinterlegt, und hohe Risikoprioritätszahlen werden im System mit einem warnenden Rot gekennzeichnet.[914]

913) Vgl. hierzu die Ausführungen in Kapitel 3.1.4.1.5, S. 51 f., sowie ergänzend VDA-Band 4 (2003), S. 24.

914) Bei einer maximal erreichbaren Risikoprioritätszahl von 1000, die erreicht wird, wenn die maximal zulässigen Werte für B, A und E von 10 jeweils erreicht werden, wurden zehn Schwellenwerte berücksichtigt, d. h. es gibt zehn Farbabstufungen bei OntoFMEA, die jeweils als kummulierte 100er-Werte der Risikoprioritätszahl gleichverteilt wurden und von gelb nach rot verlaufen (siehe hierzu auch die Abbildung 68).

Weil die hier verwendete Beispiel-FMEA (C 5960 A) der Karl Schumacher GmbH keine hohen Einzelwerte (für B, A und E) enthält, kommt es lediglich zur Anzeige von gelben Fehlertupeln in der Abbildung 68.

Für die Ermittlung der Risikoprioritätszahl wird auf eine weitere Regel in der Ontologie zurückgegriffen. In Abweichung zu den voranstehenden Regeln greift diese Regel jedoch auf ein Built-In der Inferenzmaschine OntoBroker zurück, die es erlaubt, Zahlenwerte miteinander zu multiplizieren (`X is *(A,B))`.[915] Insofern weicht die Kodierung von einer „reinen" F-Logic-Kodierung der Ontologie an dieser Stelle ab:[916]

```
FORALL W,X,Y,Z,RPZ Z:fehlertupel[hat_rpz->RPZ] <-
                       Z[hat_bedeutung->W] AND
                       Z[hat_auftrittswahrscheinlichkeit->X] AND
                       Z[hat_entdeckungswahrscheinlichkeit->Y] AND
                       evaluable_(RPZ,*(*(W,X),Y)).
```

Sofern einer Belegung der Variable `Z` durch eine Instanz des Konzepts `fehlertupel` jeweils ein Wert für die Bedeutung, die Auftrittswahrscheinlichkeit und die Entdeckungswahrscheinlichkeit zugewiesen wurde, wird durch Multiplikation der drei Werte der Variable `RPZ` das Ergebnis der Multiplikation zugewiesen. Die Variable `RPZ` wird im Kopf der Regel durch die Relation `hat_rpz` der Instanz des Konzepts `fehlertupel` zugeordnet.

Die Abbildung 68 verdeutlicht die Verwendung und Darstellung von Risikoprioritätszahlen in der Fehlerstruktur einer bis hierin durchgeführten FMEA.

---

915) Vgl. Ontoprise (2004), S. 26. Durch eine Verschachtelung der Multiplikatoren können mehr als zwei Werte miteinander multipliziert werden. Bspw. wird die Variable X mit der Instanz des Produkts (24) der Werte 2, 3 und 4 belegt, wenn der Ausdruck `(X is *(*(2,3),4))` von der Inferenzmaschine verarbeitet wird. Bei der folgend vorgestellten Regel wird eine hierzu alternative Notation verwendet, in dem für den festgelegten Vorgang das Prädikat `evaluable_` als Verwendung des Built-Ins vor den Ausdruck gesetzt wird (alternative Notation: `evaluable_(X, *(*(2,3),4)))`.

916) Als Nachteil ergibt sich hieraus eine unerklärte Semantik gemäß F-Logic, die aufgrund der Einfachheit der Anwendung, insbesondere der Kompaktheit der Formeln anstelle sehr komplizierter prädikatenlogischer Formulierungen, jedoch vernachlässigt werden kann.

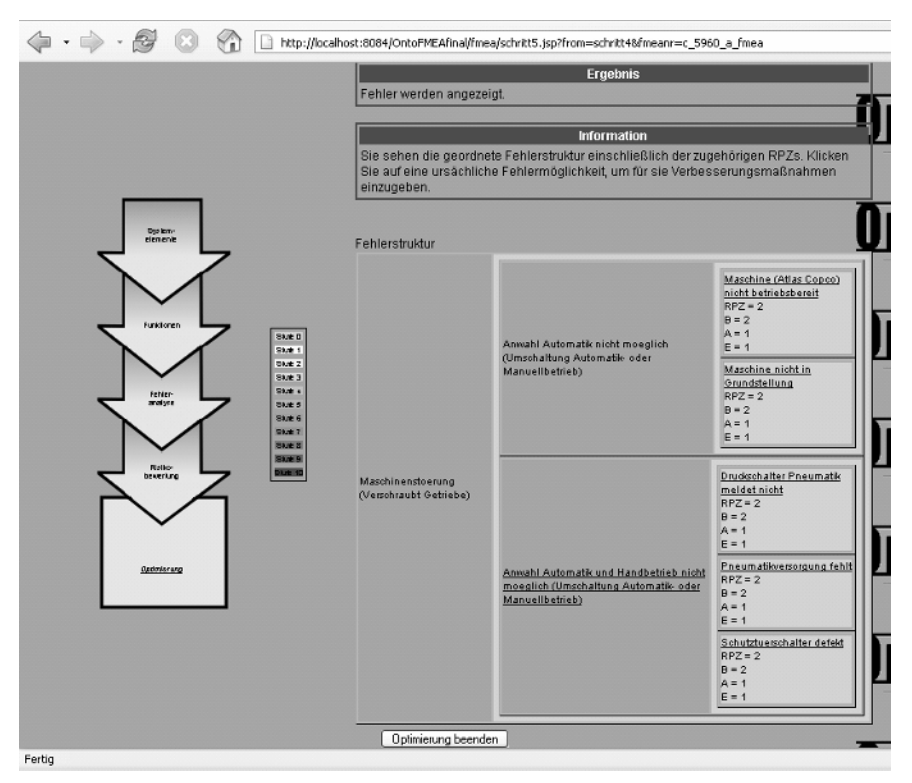


Abbildung 68: Anzeigen der Fehlfunktionsstruktur mit RPZs in OntoFMEA

Durch Anwahl der Fehlerbezeichnung in der Abbildung 68 gelangt der Benutzer zu einer Eingabemaske, die ihm das Anlegen eines weiteren „verbessernden“ Fehlertupels zu einer Fehlerursache erlaubt. Im vorliegenden Beispiel wählt der Benutzer den Fehler `Schutztuerschalter defekt` und gelangt so zur Eingabemaske, die in Abbildung 69 dargestellt wird.

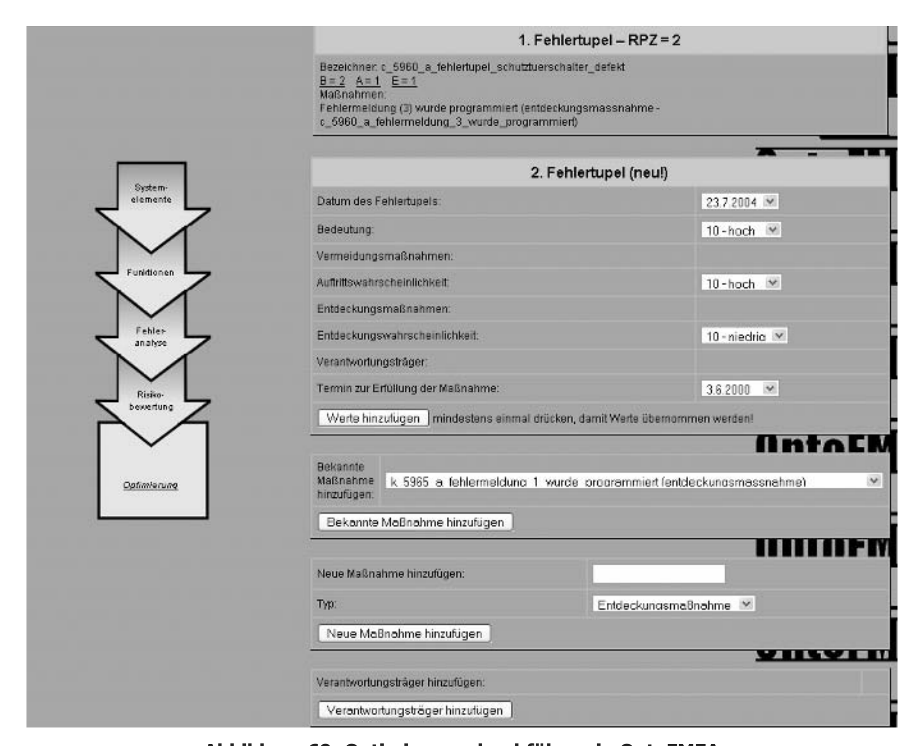


Abbildung 69: Optimierung durchführen in OntoFMEA

Innerhalb der Eingabemaske kann der Benutzer alle Informationen eingeben, die für die Festlegung eines Fehlertupels benötigt werden. Insbesondere kann der Benutzer entscheiden, ob eine bereits bekannte Maßnahme überarbeitet oder eine komplett neue Maßnahme angelegt werden soll.

Nach der Beendigung des 5. Schritts „Optimierung" zur Durchführung einer FMEA gelangt der Benutzer zurück zum Startbildschirm von OntoFMEA. Bei erfolgreicher Erstellung einer neuen FMEA erscheint diese zukünftig mittig in der Bildschirmansicht als einsehbare FMEA (siehe hierzu auch Abbildung 70) und kann über den Menüpunkt *Anzeigen erstellter FMEA* eingesehen und kontrolliert werden.

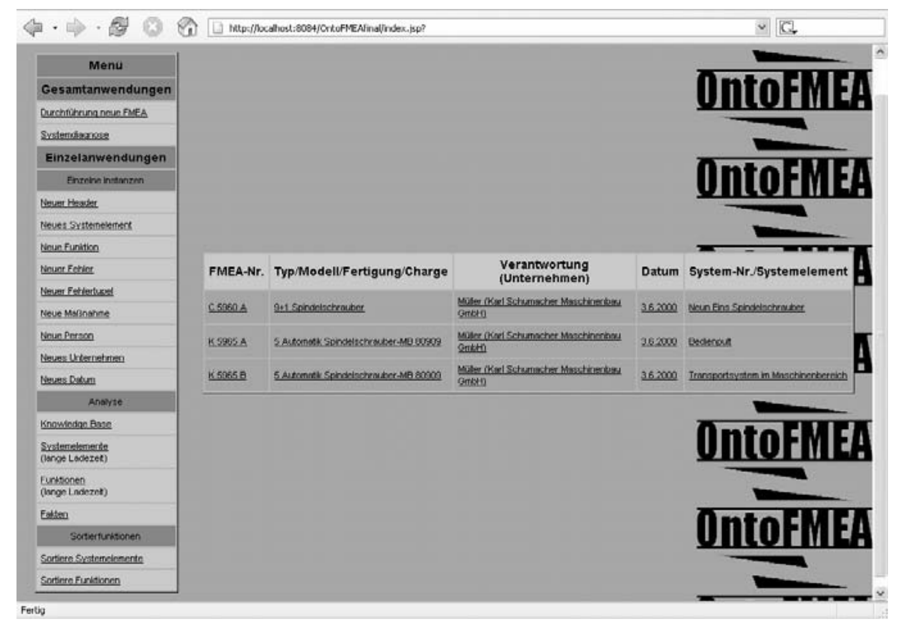


Abbildung 70: Startbildschirm OntoFMEA nach erfolgreichem Anlegen einer FMEA

Eine zweite Möglichkeit der Kontrolle der angelegten Fakten wird im Startmenü von OntoFMEA bereitgehalten. Die bei der Durchführung einer neuen FMEA erstellten Fakten lassen sich über den Menüpunkt *Fakten*[917] anzeigen. Die Darstellung erfolgt hierbei in F-Logic-Kodierung.

Über diesen Menüpunkt bietet sich dem Benutzer zum einen die Möglichkeit, die erstellten Fakten noch einmal zu kontrollieren. Zum anderen werden die Fakten während des laufenden Betriebs durch die Inferenzmaschine in die Wissensbasis zwar aufgenommen, jedoch gehen diese Fakten mit dem Herunterfahren der Inferenzmaschine verloren. Um dennoch die Informationen der „neuen" FMEA zu konservieren, kann die F-Logic-Kodierung der Fakten mittels Cut & Paste-Funktion des Betriebssystems ausgeschnitten und in die ursprüngliche FMEA-Ontologie im weiteren Sinne (in die Wissensbasis auf der Instanzenebene) eingefügt werden.[918] Somit wird das Wissen beim nächsten Start der Inferenzmaschine erneut zur Verfügung stehen.

917) Wie bereits an anderer Stelle erwähnt wurde, handelt es sich bei Fakten um quantoren- und variablenfreie, vollständig instanziierte F-Atome und F-Moleküle.

918) Sollte das Softwaresystem OntoFMEA über den derzeitigen Entwicklungszustand eines Prototyps weiterentwickelt werden, so wäre es an dieser Stelle des Softwaresystems notwendig, die Speicherung der neuen Fakten zu automatisieren. Technisch stellt diese Herausforderung einen Softwareentwickler vor nicht unüberwindbare Hindernisse.

Die Abbildung 71 verdeutlicht die Bildschirmansicht der F-Logic-Kodierung nach Anwahl des Menüpunkts *Fakten* für das behandelte Beispiel des Bedienpults eines 9+1 Spindelschraubers der Karl Schumacher Maschinenbau GmbH.



Abbildung 71: Darstellung der neu angelegten Fakten in OntoFMEA

8.4 Menüpunkt Anzeigen erstellter FMEA

8.4.1 Systemelementstruktur einsehen

Die mittig auf dem Startbildschirm aufgelisteten FMEAs können als Menüpunkt ausgewählt werden. Das Wissen einer bereits angelegten FMEA wird hier verfügbar gemacht.

Nach der Auswahl einer bestimmten FMEA wird zunächst die Systemelementstruktur angezeigt. Die grafische Benutzeroberfläche verfolgt dabei in der Regel eine dreigeteilte Darstellung des Wissens: Für den Fall der Anzeige der Systemelementstruktur findet sich in der Mitte das aktuell untersuchte Systemelement mit seinen zugeordneten Funktionen. Links vom aktuellen Systemelement befinden sich die übergeordneten und rechts davon befinden sich die untergeordneten Systemelemente.

Im Folgenden wird die Funktionalität des Menüpunkts *Anzeigen erstellter FMEA* anhand des bereits im vorigen Unterkapitel eingeführten, verkürzten Beispiels mit der FMEA-Nr. C 5960 A erläutert. Das Beispiel enthält lediglich ein oberes Systemelement (`Neun Eins Spindelschrauber`) und ein untergeordnetes Systemelement (`Bedienpult`). Für die Abbildung 72 bedeutet das, dass als aktuelles Systemelement der `Neun Eins Spindelschrauber` mit seiner zugeordneten Funktionsbeschreibung `Verschraubt Getriebe` generiert wurde. Das Systemelement wurde als Instanz des Konzepts `schraubmaschine` und die zugeordnete Funktion wurde als Instanz des Konzepts `fixieren` angelegt. Als untergeordnetes Systemelement ermittelt OntoFMEA lediglich das dem FMEA-Formblatt entsprechende `Bedienpult`.

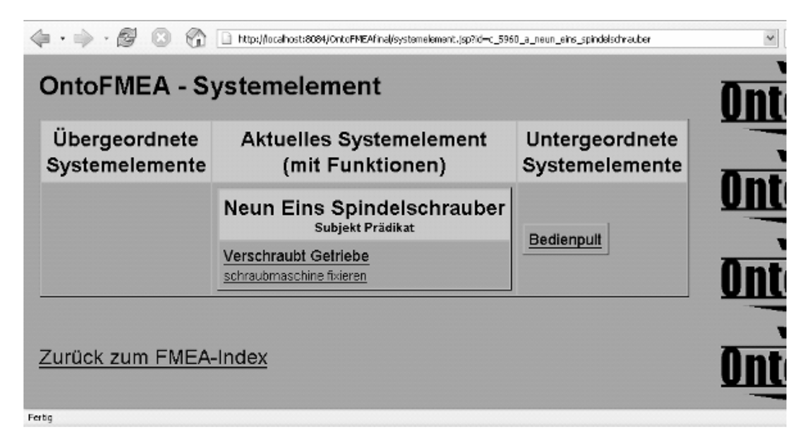


Abbildung 72: Darstellung der Systemelemente einer angelegten FMEA in OntoFMEA

Die jeweils angezeigten Systemelemente, die nicht als aktuelles Systemelement dargestellt werden, können ausgewählt und so zum aktuellen Systemelement bestimmt werden. Im vorliegenden Beispiel kann das Systemelement `Bedienpult` ausgewählt werden, und anschließend wird das Systemelement `Neun Eins Spindelschrauber` als übergeordnetes Systemelement angezeigt.

8.4.2 Funktionsstruktur einsehen

Wird die Funktion `Verschraubt Getriebe` angewählt, so wechselt die Darstellung zur Sicht auf die Funktionsstruktur (siehe Abbildung 73). In der Mitte wird die aktuell fokussierte Funktion (`Verschraubt Getriebe`) mit ihren zugeordneten Fehlermöglichkeiten (`Maschinenstoerung`) wiedergegeben; links und rechts von der aktuellen Funktion werden über- bzw. untergeordnete Funktionen (`Umschaltung Automatik- oder Manuellbetrieb`) dargestellt. Zwischen den nach links (rückwärts zur Leserichtung) gerichteten Pfeilen wird ein Link dargestellt, der es ermöglicht, wieder zurück zum zugehörigen (zugeordneten) Systemelement der aktuellen Funktion zu gelangen (`Neun Eins Spindelschrauber`).

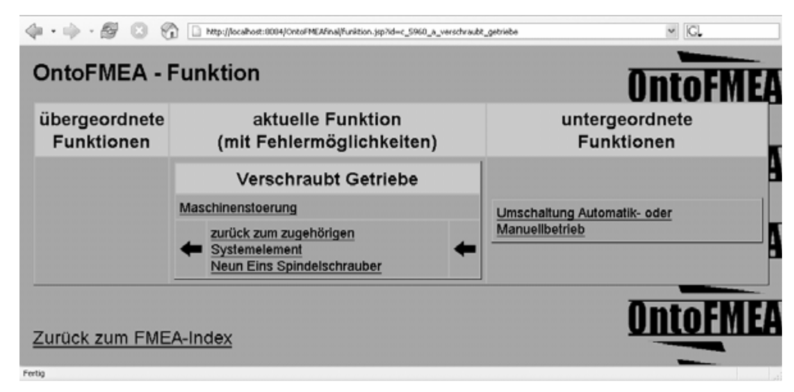


Abbildung 73: Darstellung der Funktionen einer angelegten FMEA in OntoFMEA



Abbildung 74: Wechsel der Funktion einer angelegten FMEA in OntoFMEA

Abbildung 74 verdeutlicht den Wechsel der aktuellen Funktion. Im vorliegenden Fall wurde die Funktion `Umschaltung Automatik- oder Manuellbetrieb` ausgewählt und somit in den Fokus gesetzt. Das System ermittelt in diesem Fall sieben zugeordnete Fehlermöglichkeiten für die aktuelle Funktion. Als zugehöriges Systemelement konnte das `Bedienpult` von OntoFMEA ermittelt werden. Als übergeordnete Funktion wird jetzt `Verschraubt Getriebe` ermittelt. Es konnte keine untergeordnete Funktion ermittelt werden.

8.4.3 Fehlfunktionsstruktur und Fehlertupel einsehen

Nach der Auswahl einer Fehlermöglichkeit wechselt OntoFMEA in die Fehleransicht. In der Fehleransicht werden zum einen die Fehlfunktionsstruktur und zum anderen zugeordnete Fehlertupel angezeigt. Dabei kommt es zur Anzeige von Fehlertupeln, wenn Fehlertupel für eine Fehlerursache von der Inferenzmaschine ermittelt werden können.

In Abbildung 75 wurde der aktuelle Fehler (als Fehlermöglichkeit) `Anwahl Automatik und Handbetrieb nicht moeglich` in den Fokus gestellt, und als (mögliche) Fehlerursachen wurden `Pneumatikversorgung fehlt`, `Druckschalter Pneumatik meldet nicht` sowie `Schutztuerschalter defekt` mit den jeweiligen Fehlertupeln ermittelt. Als eine mögliche Fehlerfolge wurde `Maschinenstoerung` ermittelt.[919]

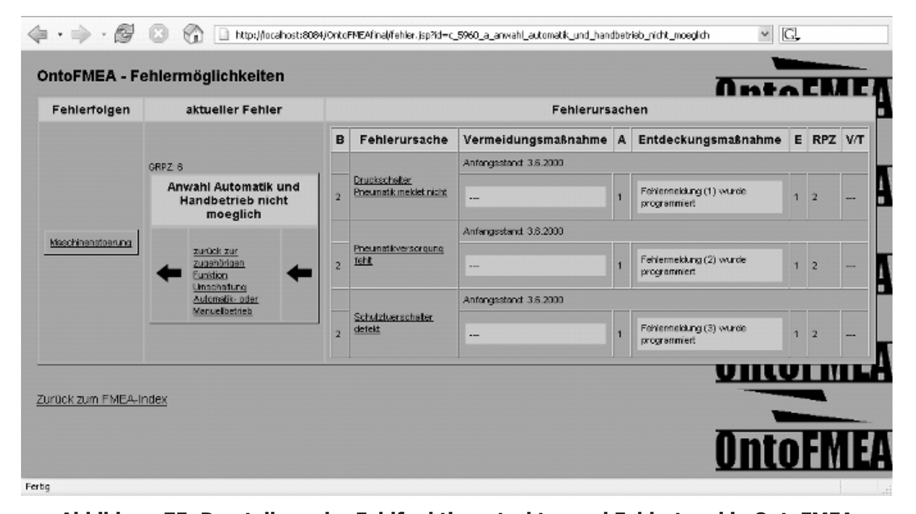


Abbildung 75: Darstellung der Fehlfunktionsstruktur und Fehlertupel in OntoFMEA

919) Hier sei noch einmal darauf hingewiesen, dass es durchaus mehrere Fehlerfolgen für einen aktuellen Fehler geben kann. OntoFMEA ermittelt immer alle möglichen Fehlerfolgen.

Die zugeordneten Werte für die Bedeutung (B) einer Fehlerfolge und für die Auftritts- (A) und Entdeckungswahrscheinlichkeiten (E) von Fehlerursachen werden bei den möglichen Fehlerursachen angegeben.[920]

Als Orientierungshilfe für den Benutzer ermittelt OntoFMEA eine Gesamtrisikoprioritätszahl (GRPZ). Die GRPZ entspricht der Addition der RPZs der ermittelbaren zugeordneten Fehlertupel eines aktuellen Fehlers.[921] Um die GRPZ zu ermitteln, greift OntoFMEA auf eine weitere Regel in der Ontologie zurück, die abermals ein Built-In der Inferenzmaschine Ontobroker berücksichtigt:[922]

---

920) An dieser Stelle lässt sich kurz eine konzeptionelle Besonderheit des hier vorgestellten Ansatzes diskutieren. Die FMEA-Ontologie erlaubt es, zu jedem Fehlertupel einen anderen, beliebigen Wert für die Bedeutung (B) einer Fehlerfolge festzulegen, d. h. die Bedeutung einer Fehlerfolge variiert je nach der zugeordneten Fehlerursache. Streng genommen sollte jedoch nur ein Wert für die Bedeutung einer Fehlerfolge aus Kundensicht angegeben werden (vgl. VDA-Band 4 (2003), S. 20), weil die Fehlerbedeutung nicht vom Fehler selbst, sondern vom ökonomischen „Wert" der Fehlerfolge abhängt. Dieser Wert sollte einzig anhand der möglichen Fehlerfolge ermittelt werden. In der Praxis findet sich jedoch auch die Variante, in der beliebige Bedeutungen für eine Fehlerfolge in Abhängigkeit von der Fehlerursache angegeben werden. Auch die Vorlagen der Karl Schumacher GmbH machen von dieser Abweichung Gebrauch. Ein Grund hierfür liegt seitens der Praxisanwender in der gemeinschaftlichen Betrachtung des Zusammenhangs zwischen der Bedeutung der Fehlerfolge und den mit ihr verbundenen Fehlerursachen. Bspw. lässt sich ein „Produktionsausfall" als Fehlerfolge auf die möglichen Fehlerursachen „Maschine zerstört" und „Maschine bekommt keinen Strom" beziehen. Hier leuchtet es unmittelbar ein, dass eine unwiderbringlich zerstörte Maschine eine höhere Bedeutung aus Kundensicht einnehmen sollte als ein vorübergehender Stromausfall. Letztlich sind solche Überlegungen in der Praxis aus der Sicht des Verfassers immer einer insuffizienten Modellierung bei der Durchführung einer FMEA geschuldet. Für das angegebene Beispiel würde dies bedeuten, dass bei eingehender Untersuchung für die beiden Fehlerursachen nicht dieselbe Fehlerfolge in Frage kommt. Bei der Durchführung der FMEA muss eine adäquate Modellierung dies berücksichtigen, bspw. indem zwei unterschiedliche Fehlerfolgen festgelegt werden („endgültiger Maschinenausfall" versus „vorübergehender Maschinenausfall").
Um dennoch die vorliegende FMEA-Ontologie in einer Weise einzusetzen, die lediglich einen einheitlichen Wert für die Bedeutung einer bestimmten Fehlerfolge aus Kundensicht sicherstellt, bietet sich – neben der Möglichkeit, zu jedem spezifischen Fehlertupel lediglich eine spezifische Fehlerfolge anzulegen, – die Formulierung einer Regel an. Exemplarisch ließe sich die folgende Regel einsetzen:

```
FORALL M,U,V,W,X,Y,Z   M[hat_fehlerbezeichnung->"Wert für Bedeutung
wurde nicht einheitlich vergeben!"] <-
          X:fehler[verursacht->>M;hat_fehlertupel->>Y] AND
          Y[hat_bedeutung->Z] AND
          V:fehler[verursacht->>M;hat_fehlertupel->>W] AND
          W[hat_bedeutung->U] AND NOT equal(Z,U).
```

Die Regel kann ohne weitere Veränderung des OntoFMEA-Prototyps eingesetzt werden. Mit Hilfe dieser Regel wird für alle Fehler, die mit Fehlertupeln verbunden sind, die nicht über dieselben Werte für die Bedeutung einer Fehlerfolge verfügen, der Hinweis in OntoFMEA ausgegeben, dass der `Wert für die Bedeutung nicht einheitlich vergeben` wurde. Die Regel nutzt das Built-In `equal(X,X)`, das die Ausprägung `true` annimmt, wenn zwei Werte gleich groß sind (vgl. Ontoprise (2004), S. 28).

921) Eine Plausibilität für die Addition der RPZ einzelner Fehlertupel ergibt sich, wenn die Fehlerursachen stochastisch unabhängig sind und man ein logisches „oder" für das Zustandekommen des aktuellen Fehlers aus den Fehlerursachen unterstellt. In der Regel werden jedoch die einzelnen RPZ einzeln betrachtet, weil bspw. die stochastische Unabhängigkeit nicht gewährleistet wird (siehe hierzu auch die Ausführungen in Kapitel 3.1.4.1.4, S. 50 f.). Das Konstrukt Gesamtrisikoprioritätszahl (GRPZ) wurde in OntoFMEA hauptsächlich eingeführt, um eine Orientierungshilfe für den Benutzer bei umfangreichen Wissensbasen zu bieten. Ein leichteres erstes Auffinden von Bereichen, die einer weiteren Untersuchung bedürfen, wird so unterstützt. Nicht zuletzt wurde die Regel auch entwickelt, um hier die Möglichkeiten von OntoFMEA weiter zu verdeutlichen.

922) Siehe hierzu auch die Regel in Kapitel 8.3.6, S. 266 ff. Zum Built-In vgl. Ontoprise (2004), S. 32.

```
FORALL X,Y,Z,RPZ,GRPZ  (Y[hat_gesamtrpz->GRPZ]) <-
            X:fehlertupel[hat_rpz->RPZ;
                         ist_fehlerursache_zugeordnet->>Z] AND
            Y[hat_ursache->>Z] AND
            sum(Y,RPZ,GRPZ).
```

Das Built-In der Inferenzmaschine OntoBroker (`sum(Y,RPZ,GRPZ)`) erlaubt es, einer Variable zugewiesene Zahlenwerte, die bspw. über eine Relation (`hat_rpz->RPZ`) zugeordnet wurden, als Instanzen eines Konzepts (Y) zu addieren und anschließend einer weiteren Variablen (GRPZ) zuzuweisen. Für die vorliegende Regel gilt: Wenn Instanzen (X) eines Fehlertupels RPZs zugewiesen wurden und diese Instanzen als mögliche Fehlerursachen einem Fehler (Y) zugeordnet wurden, dann wird diesem zugehörigen möglichen Fehler (Y) die Summe der einzelnen RPZs aus den Instanzen als Gesamtrisikoprioritätszahl (GRPZ) zugewiesen.

8.5 Menüpunkt Systemdiagnose

Mit diesem Menüpunkt wurde eine beispielhafte Anwendung für den erweiterten Einsatz von OntoFMEA und der FMEA-Ontologie – im Sinne einer Wiederverwendung des Wissens angelegter FMEAs – realisiert. Dem Benutzer wird ermöglicht, gezielt nach Ursachen für aufgetretene Fehler eines Systems zu suchen. Mögliche Einsatzgebiete sind bspw. eine Werkstatt, in der Reparaturen an defekten Systemen vorgenommen werden, und Wartungsarbeiten am Einsatzort einer eingesetzten Maschine.

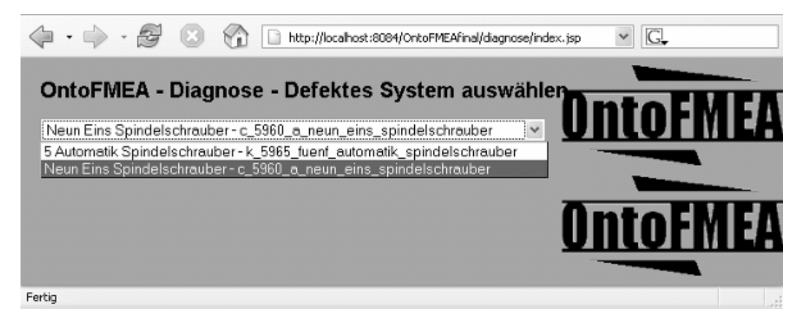


Abbildung 76: Auswahl defekter Systeme für die Systemdiagnose in OntoFMEA

Nach Auswahl des zu diagnostizierenden Systems (Abbildung 76) gelangt der Benutzer zu einer Auswahl hinterlegter Fehler des betreffenden „defekten" Systems. Für einen ausgewählten Fehler werden anschließend die möglichen Fehlerursachen ermittelt und angezeigt (Abbildung 77). Für das vorliegende verkürzte Beispiel werden zum Fehler `Anwahl Automatik und Handbetrieb nicht moeglich` die möglichen Fehlerursachen `Druckschalter Pneumatik meldet nicht`, `Pneumatikversorgung fehlt` und `Schutztuerschalter defekt` ermittelt.[923]

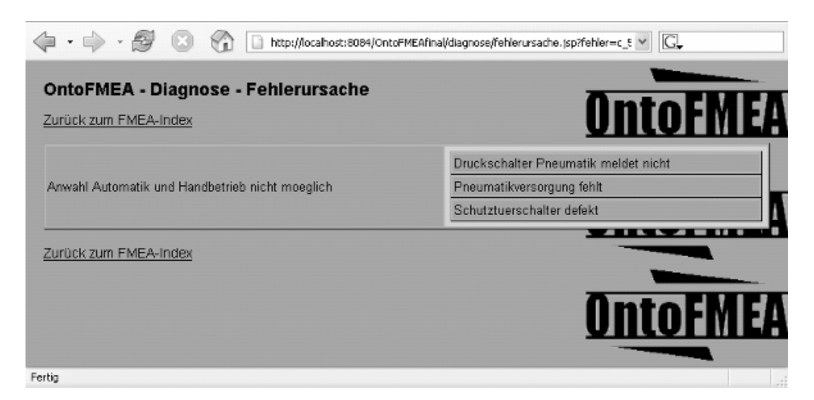


Abbildung 77: Ermittelte Fehlerursachen während der Systemdiagnose in OntoFMEA

923) Die ermittelten Fehlerursachen entsprechen somit den Fehlertupeln aus Abbildung 75, S. 274.

Der Menüpunkt *Systemdiagnose* wurde bisher nur sehr einfach in OntoFMEA realisiert. Hier sind einige Weiterentwicklungsmöglichkeiten denkbar, die an anderer Stelle dieser Arbeit diskutiert werden.[924]

924) Siehe hierzu Kapitel 10.2, S. 294 ff. Bspw. wäre es denkbar, durch weitere Regeln den Antwortraum bezüglich möglicher Fehlerursachen zu vergrößern, indem mögliche Fehlerursachen mit Hilfe einer ähnlichen Regel wie der zur Generierung möglicher Funktionen (S. 260) ermittelt werden.

9 Evaluation des integrierten Ansatzes OntoFMEA

9.1 Evaluation innerhalb des instrumentellen Umfelds

9.1.1 Untersuchungsrahmen

In diesem Kapitel wird der vorgestellte integrierte Ansatz aus OntoFMEA-Vorgehensmodell, FMEA-Ontologie und OntoFMEA-Prototyp hinsichtlich seiner möglichen Vor- und Nachteile untersucht.[925] Aus diesem Grund wird zunächst das OntoFMEA-Vorgehensmodell mit den in Kapitel 5.3.3.2, S. 170 ff., aufgestellten Anforderungen evaluiert, um die Anwendbarkeit des Vorgehensmodells für die Entwicklung von FMEA-Ontologien zu verdeutlichen. Anschließend werden „typische" Problemfelder bei der Anwendung des Instruments FMEA hinsichtlich des Lösungspotentials von Onto-FMEA (als integrierter Ansatz) diskutiert. Hieran schließt sich eine Untersuchung zur erweiterten Qualitätswirkung von OntoFMEA hinsichtlich des Potentials zur Unterstützung einer hybriden Wettbewerbsstrategie an. Abschließend wird auf weitere mögliche Anwendungsfelder von OntoFMEA exemplarisch eingegangen, um mögliche weitere Vorteile zu skizzieren.

9.1.2 Vorgehensmodell

Die Evaluation des OntoFMEA-Vorgehensmodells wird anhand der Kriterien *Generizität*, *Anwendungsbezogenheit*, *Dokumentation, Einfachheit, Klarheit* und *Werkzeugunterstützung* vorgenommen.[926] Sie basiert hauptsächlich auf den Erfahrungen aus der Anwendung dieses Vorgehensmodells zur Entwicklung der FMEA-Ontologie.

Generizität: Das Vorgehensmodell ist insofern generisch, als es eine allgemeine Beschreibung der Phasen zur Ontologiekonstruktion umfasst und für verschiedene Projekte angewendet werden kann. Es wird zur Vorbedingung gemacht, dass die zu entwickelnde Ontologie für den Einsatz in einem FMEA-System konzipiert werden soll. Das Vorgehensmodell selbst jedoch ist nicht projektspezifisch aufgebaut. Dieses Gestaltungsprinzip ermöglicht eine leichtere Anpassung an die Gegebenheiten eines konkreten Projekts und an nicht vorhersehbare Einschränkungen, die sich aus den Umgebungsbedingungen eines konkreten Projektumfelds ergeben (bspw. Ressourcenrestriktionen). Außerdem bildet dieses Gestaltungsprinzip die Voraussetzung für eine spätere Wiederverwendung des Vorgehensmodells in einem anderen Kontext, wie zum Beispiel der Entwicklung einer Ontologie für das Quality Function Deployment (QFD).

Anwendungsbezogenheit: Zusätzlich zur generischen Beschreibung und Modellierung der Phasen und ihres Ablaufs wurde eine detaillierte Erläuterung der einzelnen Aktivitäten gegeben. Diese Darstellung der Aktivitäten in den verschiedenen Phasen umfasst rudimentär auch

925) Im Folgenden wird der integrierte Ansatz als Instrument bestehend aus Vorgehensmodell, Ontologie und Prototyp verkürzt als „OntoFMEA" bezeichnet.

926) Um eine Vergleichbarkeit mit den vorgestellten Ansätzen aus Kapitel 5.1.2.3.1, S. 136 ff., zu gewährleisten, wird hier ebenfalls auf das Kriterium der Vollständigkeit verzichtet (siehe hierzu insbesondere Fn. 706, S. 172).

Empfehlungen für Methoden, die bei der Realisierung einer FMEA-Ontologie eingesetzt werden können. Für die Thematik „FMEA“ wurde ein Ebenen-Konzept implementiert, das die Entwicklung einer FMEA-Ontologie strukturiert und erleichtert. Diese Empfehlungen sollen eine Hilfestellung für die Ontologieentwickler bieten und die Anwendbarkeit des Vorgehensmodells fördern.

Dokumentation: Damit ihre Bedeutung als Bestandteil des Entwicklungsprozesses explizit dargestellt wird, bildet die Dokumentation eine eigene Phase. Darüber hinaus wird in jeder Phase das Festhalten von Entscheidungen und deren Begründungen sowie der jeweiligen Meilensteine und schriftlichen Ergebnisse berücksichtigt (z. B. in Anforderungsspezifikation, Konzeptualisierung, Projektplan und Wissensträgerkarte). Dadurch enthält das Vorgehensmodell eine durchgängige Dokumentation während der gesamten Ontologiekonstruktion und verbessert damit die Nachvollziehbarkeit des Prozesses sowohl während seiner Durchführung als auch im Nachhinein. Außerdem bilden die Dokumentation der Entwicklung und die Begründung der wichtigen Entscheidungen eine „Erweiterung“ des Vorgehensmodells selbst, da die Wiederverwendung der gesammelten Erfahrungen bei einem später eventuell stattfindenden ähnlichen Projekt von großem Vorteil sein kann.

Einfachheit: Die Inhalte der Phasen sind detailliert, aber nicht komplex beschrieben und auf eine nicht-technische Art formuliert, um grundsätzlich für alle am Projekt mitwirkenden Personen verständlich zu sein. Diese Eigenschaften machen die Simplizität des Modells aus; sie sind wichtig, damit die Beteiligten auch zu einer tatsächlichen Anwendung des Vorgehensmodells motiviert sind. Ein zu komplexes Vorgehensmodell wird möglicherweise nicht akzeptiert und deswegen nicht eingesetzt; damit verliert es seinen Nutzen.

Klarheit: Das Vorgehensmodell besitzt eine klare Gliederung und umfasst wenige Phasen, was insbesondere durch den grafischen Überblick (siehe Abbildung 43, S. 199) veranschaulicht wird. Auch der Grundsatz der Klarheit besitzt eine große Bedeutung für die Anwendbarkeit eines Vorgehensmodells, da Mehrdeutigkeiten zu einer „falschen“ (nicht intendierten) Umsetzung oder sogar zu einer Ablehnung der Anwendung führen können. Aus diesem Grund wurden die Phasenbeschreibungen möglichst präzise formuliert. Hierzu trägt besonders die semi-formale Darstellung durch Aktivitätsdiagramme bei. Diese Modellierungsart ermöglicht eine präzise und anschauliche Abbildung des Phasenablaufs und bildet zugleich eine wichtige Ergänzung zur informalen textuellen Beschreibung, da die einzelnen Aktionen und Informationsträger benannt und zueinander in Beziehung gesetzt werden.

Werkzeugunterstützung: Die Nutzung von Informationstechnologie in einem Entwicklungsprojekt für Software ist vorteilhaft für eine leichtere und effizientere Prozessdurchführung, weil zum Beispiel große Informationsmengen verwaltet und einzelne Aktionen sogar automatisiert werden können. Daher werden im vorgestellten Vorgehensmodell nicht nur mögliche Methoden zur Umsetzung genannt, sondern für einzelne Phasen auch entsprechende Computerwerkzeuge zur Erleichterung und Automatisierung der Aktivitäten vorgeschlagen.

Für die Konstruktion von FMEA-Ontologien sind zur Zeit noch keine Programme verfügbar, die alle Phasen des Entwicklungsprozesses gleichermaßen unterstützen, doch es gibt bereits

zahlreiche Ontologieentwicklungsumgebungen zur Erstellung, Änderung, Visualisierung, Annotierung und auch zur Überprüfung von Ontologien (wie etwa die in den Kapiteln 4.1.4.3.1, S. 91 f., vorgestellten Produkte OntoEdit, Protégé-2000 und WebODE). Diese können im Rahmen der Konzeptualisierung und Formalisierung – eventuell auch bei der Evaluation und Dokumentation – eingesetzt und durch Computerwerkzeuge aus den Bereichen Software Engineering und Knowledge Engineering (bspw. Anforderungsmanagement- und Dokumentations-Tools) ergänzt werden. Die im Vorgehensmodell beschriebenen Werkzeuge erleichtern die Ontologieentwicklung für die Projektbeteiligten und führen damit zu einer höheren Anwendbarkeit und Akzeptanz des Vorgehensmodells.

9.1.3 Prototyp und Ontologie

Aus der herkömmlichen Verwendung des Instruments FMEA ergeben sich Problembereiche, die einer erfolgreichen Verwendung oftmals im Wege stehen. Es finden sich im Einzelnen bei der Verwendung die folgende Problembereiche (siehe auch Abbildung 78):[927]

1. Das *Wissensbereitstellungsproblem* erstreckt sich auf die oftmals eingeschränkte Verfügbarkeit von qualifizierten Experten und deren Motivation, Wissen preiszugeben, d. h., die Verfügbarkeit von Wissen ist nicht immer gewährleistet.

2. Das *Wissensverarbeitungsproblem* erstreckt sich auf die unterschiedlichen Wissensarten, die zu repräsentieren sind (Erfahrungswissen, statistische Daten usw.), um sie einer Verarbeitung zuzuführen, und auf die interdisziplinäre Wissensverarbeitung (Teamarbeit mit Mitgliedern unterschiedlicher Fachgebiete)[928].

3. Das *Komplexitätsproblem* erstreckt sich auf den hohen Zeitaufwand und den hohen Schwierigkeitsgrad einer FMEA-Durchführung.

4. Das *Integrationsproblem* erstreckt sich auf die Schaffung eines durchgängigen Qualitätswesens, das auch weitere Bereiche integriert, wie z. B. die Kosten- und die Arbeitsplanung oder die Berichterstellung.

5. Das *Suchproblem* erstreckt sich vornehmlich auf die Ermittlung geeigneter Suchkriterien, um die Wiederverwendung von vorhandenem FMEA-Wissen zu ermöglichen.

6. Das *Aktualisierungsproblem* erstreckt sich auf die Verfügbarkeit von FMEA-Wissen des jeweils aktuellsten Stands: es muss sichergestellt werden, dass kein veraltetes Wissen wiederverwendet wird.

---

927) Vgl. Nedeß, Nickel (1992), S. 311 f.

928) Die Zusammenkunft von Mitgliedern unterschiedlicher Fachgebiete impliziert sprachliche Barrieren aufgrund der unterschiedlichen Wissenshintergründe dieser Mitglieder, die erst noch durch aufwändige Abgleichungsprozesse angepasst werden müssen.

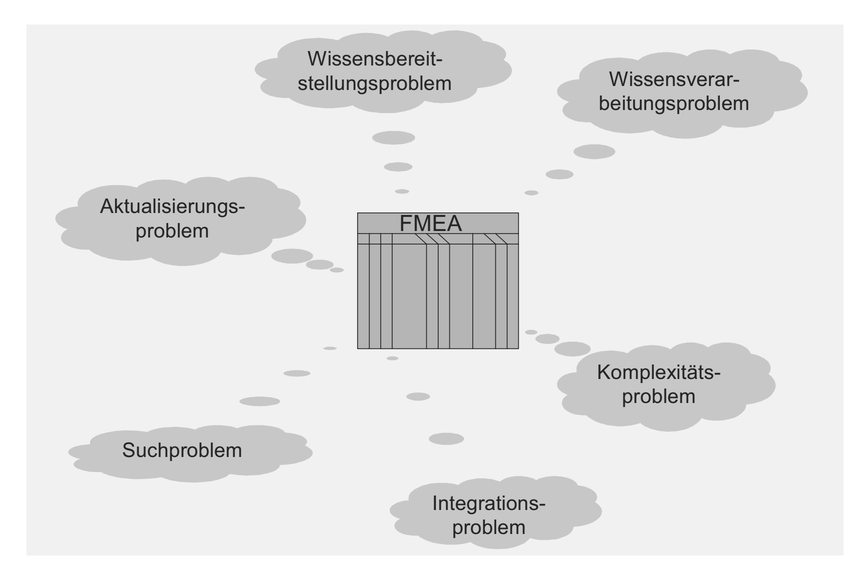


Abbildung 78: Problembereiche einer FMEA

Nedeß und Nickel weisen darauf hin, dass sich diese Problembereiche trotz der vorhandenen IT-gestützten Werkzeuge ergeben. Sie räumen einem Wissensbasierten System die größten Chancen zur Verbesserung dieser Problemsituation ein.[929]

Das Potential zur Verbesserung der einzelnen Problembereiche ergibt sich, wie folgt, für den Einsatz von OntoFMEA. Es wird hierbei jeweils kurz auf die Vorteile von OntoFMEA eingegangen. Dabei werden – wo dies notwendig erscheint – als Referenzsysteme herkömmliche Datenbanksysteme berücksichtigt. Die Argumentation bezieht sich vornehmlich auf die Anwendung des OntoFMEA-Prototyps, dessen wesentlicher Kern die FMEA-Ontologie darstellt:

1. Die *Wissensbereitstellung* wird über den OntoFMEA-Prototyp verbessert, indem die Verfügbarkeit des Wissens jederzeit gewährleistet wird. Je mehr Wissen in der Wissensbasis enthalten ist, desto weniger Wissen der Experten wird benötigt. Die Abhängigkeit von der Verfügbarkeit von Experten wird gemindert. Durch den größeren Antwortraum, der mit einer Inferenzmaschine erreicht werden kann, erhält der Benutzer tendenziell mehr für ihn relevantes Wissen als in einem herkömmlichen FMEA-System. Das OntoFMEA-Vorgehensmodell erleichtert durch sein detailliertes Vorgehen die Wissensakquisition. Experten werden bei einem eindeutig strukturierten Vorgehen tendenziell eher dazu motiviert, ihr Wissen preiszugeben, weil für sie nachvollziehbar wird, was damit geschehen wird.

2. In der *Wissensverarbeitung* mit dem OntoFMEA-Prototyp ist bei der Wissensauswertung die Verarbeitung von unsicherem und unvollständigem Wissen möglich, d. h., es werden verschiedene Wissensarten verarbeitet. Weil Ontologien grundsätzlich auf ei-

929) Vgl. Nedeß, Nickel (1992), S. 314.

nem gemeinsam wahrgenommenen Realitätsausschnitt basieren, wird die interdisziplinäre Wissensverarbeitung erleichtert oder sogar erst ermöglicht. Das Wissen in der Wissensbasis kann zusätzlich von Akteuren aus unterschiedlichen Anwendungsbereichen (z. B. Vertrieb und Konstruktion) genutzt werden.[930]

3. Bei zunehmender *Komplexität* steigen mit dem Wissen die zu verarbeitenden Datenbestände an. Herkömmliche Datenbanksysteme haben hier mit zunehmender Komplexität gegenüber deduktiven objektorientierten Systemen zeitliche Vorteile. Die Inferenzmaschine OntoBroker erlaubt die Anbindung von herkömmlichen Datenbanken, so dass sich der zeitliche Vorteil eingeschränkt auch auf den Onto-FMEA-Prototyp übertragen lässt.

4. Über die FMEA-Ontologie ist eine *Integration* weiterer Bereiche des Qualitätswesens möglich.[931] Hier liegt ein Hauptvorteil in der Verwendung von Ontologien beim Einsatz von Wissensbasierten Systemen.[932] Gelingt es dem Entwicklungsteam, eine klare Ontologie zu entwickeln, so bleiben alle Vorteile (einschließlich des Vorteils zur Komplexität) erhalten.

5. Die *Suche* von vorhandenem FMEA-Wissen zur Wiederverwendung kann durch den OntoFMEA-Prototyp auf zwei Arten verbessert werden. Zum einen werden mögliche Fehler (als Beispiele) direkt bei der FMEA-Durchführung vorgeschlagen.[933] Zum anderen lässt sich die Wissensbasis, bspw. anhand der hierarchischen taxonomischen Relationen, durchforsten. Eine solche Baumstruktur vereinfacht das Wiederauffinden von FMEA-Wissen wesentlich.

6. Das abgeleitete Wissen während einer Anwendung steht dem Wissensbasierten System unmittelbar zur Verfügung. Eine *Aktualisierung* der Wissensbasis erfolgt somit „on the fly“, d. h. es ist permanent sichergestellt, dass kein veraltetes Wissen – von dem in irgendeiner Form bekannt ist, dass es veraltet ist, – verwendet wird.

930) Siehe hierzu auch das Beispiel „Kompetenzmanagementsystem“ in Kapitel 9.3.1, S. 287 f.

931) Siehe hierzu beispielhaft Kapitel 9.3.2, S. 288 ff.

932) Siehe hierzu auch das folgende Kapitel.

933) Siehe hierzu Kapitel 8.3.4, S. 261 ff.

9.2 Evaluation innerhalb des wettbewerblichen Umfelds

Es lassen sich Effekte durch den Einsatz des Instruments OntoFMEA (als integrierter Ansatz)[934] erzielen, die eine Auswirkung auf die Qualität und damit simultan auf die beiden generischen Wettbewerbsstrategien Porter´s (Kostenführerschaft und Qualitätsführerschaft [als Ausprägung der Differenzierung]; siehe hierzu Abbildung 1, S. 6) haben.[935] Die Diskussion des Ansatzes von OntoFMEA zur Unterstützungsmöglichkeit einer hybriden (simultanen) Wettbewerbsstrategie wird im Folgenden anhand der Erweiterung der Abbildung 1, S. 6, zur Darstellung der Wirkungsweise von Qualität geführt (siehe Abbildung 79, S. 286). Es wird davon ausgegangen, dass *Onto-FMEA* auf der instrumentellen Ebene zum Einsatz gelangt:

- Die Anwendung von OntoFMEA als integriertes Instrument ermöglicht, dass dem Mitarbeiter als Benutzer von OntoFMEA Erfahrungswissen als *strukturiertes, gesammeltes Wissen* verfügbar gemacht wird. Durch kontinuierliche Anwendung des OntoFMEA-Prototyps wird dieses Wissen stetig vergrößert.

- Die *Fehleranzahl* bei der Konstruktion und Entwicklung neuer Produkte wird durch systematische Verhinderung von Fehlerursachen, Fehlern und Fehlerfolgen bei der Anwendung des OntoFMEA-Prototyps kontinuierlich verringert. Zum einen wird die Fehleranzahl direkt durch die Verwendung des Wissensbasierten Systems gesenkt, weil das methodische Vorgehen bspw. Flüchtigkeitsfehler ausschließt. Zum anderen vermeidet gerade die Hinzunahme des gesammelten, strukturierten Erfahrungswissens, welches sich im Laufe der Anwendung stetig vergrößert, bspw. Wiederholungsfehler. In diesem Sinne vermeidet Wissen Fehler.

- Die verringerte Fehleranzahl wirkt sich positiv sowohl auf die *technische Qualität* als auch auf die *wahrgenommene Qualität* aus. Eine erhöhte relative Effizienz bei der Erfüllung von Spezifikationen führt zu einer Reduzierung der Qualitätskosten und damit zu einer Senkung der Gesamtkosten. Die vom Kunden wahrgenommene relative Qualität wird gesteigert, wenn einem Produkt weniger tatsächliche Fehler als einem vergleichbaren Alternativprodukt innewohnen. Dies führt zu einer erhöhten Preisbereitschaft seitens der Kunden, so dass höhere Absatzpreise durchgesetzt werden können, die wiederum die Rentabilität steigern. Voraussetzung hierfür ist, dass es dem Unternehmen gelingt, den Qualitätsvorsprung zu kommunizieren und dass der Kunde überhaupt bereit ist, höhere Qualität zu bezahlen.

- Durch die Wiederwendung von strukturiertem, gesammeltem (Erfahrungs-)Wissen und die Senkung der Fehleranzahl wird eine *verkürzte Entwicklungszeit* realisiert, weil Doppelarbeiten vermieden werden und weniger Änderungen während des Entwicklungsprozesses durchgeführt werden müssen.[936]

934) Siehe auch Fn. 925, S. 279.

935) Zur erläuternden Argumentation bezüglich des indirekten Einflusses von Qualität auf den Faktor Kosten siehe Kapitel 1.1.2.2, S. 3 ff.

936) Vgl. Kamiske, Brauer (2003), S. 75.

Zum einen wirkt sich eine verkürzte Entwicklungszeit positiv auf die relativen Kosten, die Kosten im Verhältnis zum üblicherweise stärksten Wettbewerber, aus, weil weniger Personenstunden zu Buche schlagen (bei gleichbleibender Kapazität pro Zeiteinheit). Zum anderen wirkt sich eine verkürzte Entwicklungszeit positiv auf den relativen Marktanteil des Unternehmens aus, weil durch den technologischen Vorsprung auch ein zeitlicher Vorsprung vor den Wettbewerbern genutzt werden kann, um einen Markt zu bedienen. Des Weiteren lässt sich das Nutzen-Preis-Verhältnis für den Kunden steigern, indem das Produkt aufgrund geringerer Gesamtkosten (die sich auch aus einer kürzeren Entwicklungzeit ergeben) zu einem geringeren Preis angeboten werden kann.

Aufgrund dieser erweiterten Qualitätswirkungen des integrierten Instruments OntoFMEA lässt sich feststellen, dass es mit Hilfe von OntoFMEA simultan möglich ist, sowohl die generische Strategie der Kostenführerschaft als auch die generische Strategie der Qualitätsführerschaft zu unterstützen.

Die Abbildung 79 auf der nächsten Seite fasst diesen Sachverhalt zusammen. Die gestrichelten Pfeile verdeutlichen dabei die Auswirkungen der instrumentellen Ebene auf die wettbewerbliche Ebene. Die durchgezogenen Pfeile verdeutlichen die Auswirkungen der einzelnen Faktoren innerhalb einer Ebene. Die Faktoren werden jeweils in abgerundeten Rechtecken erfasst und gegebenenfalls zu Sinneinheiten (nicht abgerundete Rechtecke) zusammengefasst. Die vier roten Faktoren vertreten in diesem Ausschnitt das instrumentelle Umfeld. Die restlichen Faktoren werden dem wettbewerblichen Umfeld zugesprochen.

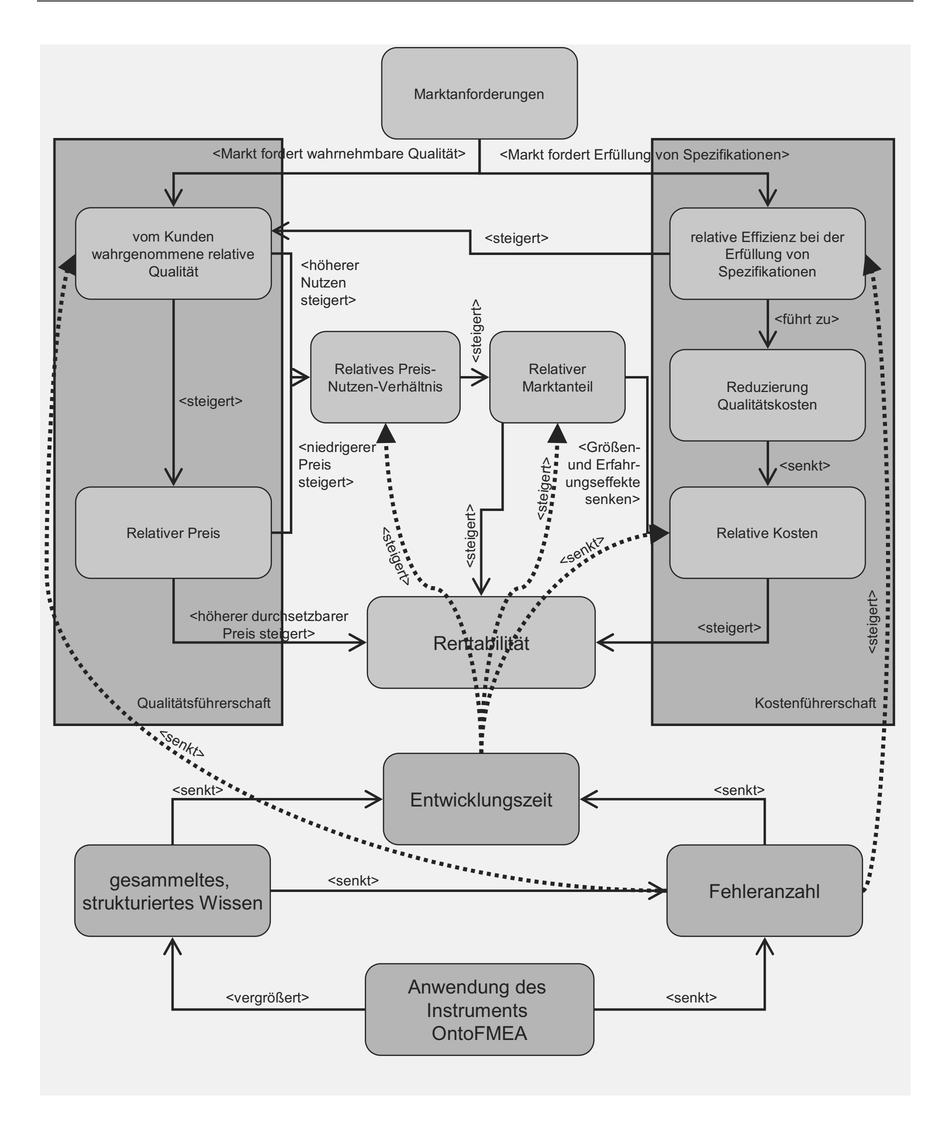


Abbildung 79: Erweiterte Wirkung der Qualität durch OntoFMEA

9.3 Anwendungsnähe des integrierten Ansatzes OntoFMEA

Um den Nutzen weiter zu skizzieren, der mit OntoFMEA erreicht werden kann, wird in diesem Kapitel auf einige ausgewählte Anwendungsbereiche für Erweiterungen des OntoFMEA-Ansatzes eingegangen. Dieses Kapitel dient somit der qualitativen Veranschaulichung des Nutzens in Bezug auf die Anwendungsnähe und Praktikabilität des vorgestellten integrierten Ansatzes. Auch bereitet dieses Kapitel das Kapitel 10.2, S. 294 ff., zum weiteren Forschungsbedarf vor.

In diesem Zusammenhang werden drei kurze Beispiele, die sich anhand ihres Einsatzzwecks unterscheiden lassen, vorgestellt. Erstens wird ein Beispiel für die Integration zweier ontologiebasierter Systeme gegeben. Zweitens wird ein Beispiel für die Erweiterung der Ontologie auf weitere Bereiche des Qualitätsmanagements gegeben. Drittens wird ein Beispiel für eine erweiterte Nutzung des im System enthaltenen Wissens gegeben.

9.3.1 Kompetenzmanagementsystem

Während der letzten Jahre erkennen mehr und mehr Unternehmen das Wissen über die zur Verfügung stehenden Mitarbeiterkompetenzen als einen strategischen Erfolgsfaktor an.[937] Um die Wettbewerbsfähigkeit eines Unternehmens zu sichern, müssen bspw. die Kompetenzen von ausscheidenden Mitarbeitern ersetzt werden.[938] Hierzu gelangen vermehrt computergestützte Kompetenzmanagementsysteme zum Einsatz. Herkömmliche Systeme konzentrieren sich auf die manuelle Eingabe von Wissen (zumeist vorgenommen durch die Kompetenzträger selbst), eine simple Baumhierarchie in der Darstellung der Beziehungen zwischen unterschiedlichen Kompetenzen und den Einsatz des Kompetenzmanagementsystems in der Personalbteilung (von zumeist großen Unternehmen).[939] Das *Managen von Wissen über Kompetenzen* wird dabei in der Hauptsache von zwei Problemfeldern beeinflusst.

Zum einen lässt sich das Wissen über Mitarbeiterkompetenzen selten in expliziter Form (wie etwa spezifischen Softwareanwendungen einer Personalabteilung) identifizieren. Vielmehr findet sich das Wissen über Mitarbeiterkompetenzen in der Regel in Quellen, die sich nur mittelbar für das Kompetenzmanagement mit dem Computer erschließen lassen, wie etwa QM-Handbüchern oder FMEA-Durchführungen. Hinzu kommt, dass die vorgefundenen Quellen oftmals auf heterogenen Sprachwelten aufsetzen. Diese heterogenen Sprachwelten führen beim Versuch, die einzelnen Quellen synchron zu nutzen, zu dem Problem, dass sie erst in eine gemeinsam verstandene Sprachwelt übertragen werden müssen.

Das Wissen aus bereits durchgeführten FMEAs kann genutzt werden, um die Wissensbasis eines ontologiebasierten Kompetenzmanagementsystems zu vergrößern und damit die Anwendbarkeit eines solchen Systems zu erhöhen. Hierzu wird insbesondere das Wissen über die Verantwortlichkeit eines Mitarbeiters bezüglich einer festgelegten Maßnahme als Wissen

937) Vgl. Bartlett, Ghoshal (2002), S. 34; Knaese, Probst (2001), S. 35.

938) Vgl. Raphael (2002), S. 48.

939) Vgl. Maedche (2002), S. 4.

über dessen Kompetenzen berücksichtigt. Folgende Inferenzregel soll hier als Beispiel zur Verdeutlichung dienen:[940]

```
FORALL A,B,C,D,X,Y    X[hat_kompetenz->>Y]<-
                      X:person AND Y:systemelement AND
                      A:massnahme[wird_angewendet->>B] AND
                      B:fehlertupel[ist_verantwortlich->>X;
                        ist_fehlerursache_zugeordnet->>C]AND
                      C:fehler[verhindert_funktion->>D] AND
                      D:funktion[ist_funktion_von_systemelement->>Y].
```

Wenn eine Person verantwortlich ist für die Durchführung einer Maßnahme aufgrund ihrer Verantwortlichkeit für ein Fehlertupel zur Fehlerbearbeitung und diese Maßnahme ein zugehöriges bestimmtes Systemelement betrifft, so wird der Person hinsichtlich des Umgangs mit dem Systemelement eine Kompetenz zugeschrieben.

Durch diese Regel wird dem ontologiebasierten Kompetenzmanagementsystem ermöglicht, die Wissensbasis um Inhalte aus der FMEA-Durchführung zu erweitern. Der Anwender des Kompetenzmanagementsystems erhält erweitertes Wissen über die Kompetenzen von Mitarbeitern, die es ihm ermöglichen, Kompetenzträger gezielter zu ermitteln.[941]

9.3.2 Integration weiterer Instrumente des Qualitätsmanagements

Ein weiteres Anwendungsbeispiel für eine Erweiterung des OntoFMEA-Ansatzes liegt in der Integration zusätzlicher Instrumente des Qualitätsmanagements. Hierzu bietet sich insbesondere das Quality Function Deployment (QFD)[942] an, weil es eine „natürliche" Schnittstelle zur FMEA bereits vorsieht, wie im Folgenden kurz erläutert wird.

Bei der Ermittlung der Bedeutung der Folgen eines Fehlers aus der Sicht des Kunden kann auf das hinterlegte Wissen aus Durchführungen des Instruments Quality Function Deployment zurückgegriffen werden. Zentrales Element des Quality Function Deployment ist ein Matrix-Diagramm. In dem Matrix-Diagramm werden Anforderungen (WAS?) an ein zu entwickelndes Produkt und (kritische) Merkmale (WIE?) des zu entwickelnden Produkts einander gegenübergestellt, um aus den unterschiedlich starken Wechselbeziehungen Prioritäten für

940) Die Regel dient an dieser Stelle als exemplarische Ausführung in der Hauptsache der Verdeutlichung. Diskussionswürdig wäre der Umstand, ob sich eine Kompetenz auf ein Systemelement erstrecken kann oder ob lediglich bspw. die Konstruktion, Reparatur oder Wartung des Systemelements eine Kompetenz darstellt. Im letzteren Fall müsste die Regel komplexer formuliert werden.

941) Für einen tieferen Einblick zum Themengebiet des ontologiebasierten Kompetenzmanagements vgl. Dittmann, Zelewski (2004) und Zelewski, Alan et al. (2005).

942) Vgl. zu QFD Gogoll (2000), S. 363 ff., und Pfeifer (2002), S. 303 ff.

die Umsetzung des Produkts abzuleiten.[943] Üblicherweise werden vier Phasen bei der Durchführung des Quality Function Deployment unterschieden:[944]

1. Phase:
   Die Anforderungen der Kunden (WAS?) werden in technische Merkmale des Produkts übersetzt (WIE?).

2. Phase:
   Besonders beachtenswerte Merkmale als Anforderungen an das Produkt (WAS?) werden in Merkmale der Baugruppen und Komponenten etc. übersetzt (WIE?).

3. Phase:
   Besonders beachtenswerte Merkmale der Baugruppen und Komponenten etc. als Anforderungen (WAS?) werden in Merkmale der Bearbeitungsprozesse übersetzt (WIE?).

4. Phase:
   Anhand der kritischen Anforderungen an die Bearbeitungsprozesse (WAS?) werden die Fertigungs- und Prüfmittel ausgelegt (WIE?).

Die ermittelten technischen Merkmale beschließen die 1. Phase des QFD-Ansatzes. In der folgenden 2. Phase sollen die technischen Merkmale in Baugruppen und Komponenten übersetzt werden. Bereits hier deutet sich an, dass die so ermittelten Systemelemente „automatisch“ für die FMEA-Durchführung bereitgestellt und unmittelbar einer FMEA-Durchführung unterzogen werden könnten.

Eine weitere Integrationsmöglichkeit bietet die Berücksichtigung der Kundenanforderungen für die Ermittlung der Bedeutung eines Fehlers.

Das QFD-Wissen kann mittels Regeln der FMEA zur Verfügung gestellt werden. Exemplarisch gilt zur Veranschaulichung die folgende Regel:

```
FORALL A,B,C,X,Y  X:fehlertupel[hat_moegliche_bedeutung->>Y]<-
                  X:fehlertupel[ist_fehlerursache_zugeordnet->>A] AND
                  A:fehler[verhindert_funktion->>B] AND
                  B:funktion[ist_funktion_von_systemelement->>C] AND
                  C:systemelement[hat_kundenanforderungsbedeutung->>Y].
```

Die Regel ließe sich für den Fall anwenden, dass eine neue Zeile gemäß FMEA-Formblatt bei einer FMEA-Durchführung angelegt werden soll. Das Softwaresystem könnte in diesem Fall einen Vorschlag für die Bedeutung ermitteln, sofern eine entsprechende QFD-Ontologie existiert, auf die zurückgegriffen werden kann. Mit anderen Worten: Für den Fall, dass aufgrund der zusätzlichen Berücksichtigung von QFD-Wissen eine Instanz (INTEGER-Wert) für die

943) Vgl. Gogoll (2000), S. 366.

944) Vgl. Pfeifer (2002), S. 305.

Relation `hat_kundenanforderungsbedeutung`, die einer QFD-Ontologie entstammen würde, für eine Instanz des Konzepts `systemelement` ermittelt werden kann, so wird eine mögliche Instanz für die Bedeutung einer Fehlerfolge ermittelt.

Denkbar wäre, dass mittels einer weiteren Regel (nicht kodiert dargestellt) sichergestellt würde, dass nicht bereits eine Instanz für die Bedeutung einer bestimmten Fehlerfolge existiert. Sofern kein Wert existiert, ließe sich die Relation `hat_moegliche_bedeutung` zu `hat_bedeutung`, die bereits in der FMEA-Ontologie vorhanden ist, transformieren und weiterverwenden.

9.3.3 Ausbau Diagnosefunktion

Die Diagnosefunktion aus Kapitel 8.5, S. 277 f., des OntoFMEA-Prototyps kann weiter ausgebaut werden, um die Wiederverwendung von Wissen Nutzen stiftend zu forcieren. Zurzeit berücksichtigt die Diagnosefunktion vornehmlich eine Rückwärtsverkettung von einer Fehlerfolge hin zu einer Fehlerursache, d. h., es wird lediglich deduktives Schlussfolgern eingesetzt. Mit Hilfe non-deduktiver Regeln kann die Anwendbarkeit der Diagnosefunktion signifikant erweitert werden. Hierzu müssten vor allem Regeln berücksichtigt werden, die Erfahrungswissen, das die einzelnen Fehlerketten (Fehlerursache-Fehler-Fehlerfolge) miteinander verknüpft, implementieren. Beispielhaft sei hierfür genannt:

```
FORALL A,B,C,X,Y  X:systemelement[hat_moegliche_fehlerursache->>Y]<-
                  X:systemelement[hat_schnittstelle_mit->>A] AND
                  B:funktion[ist_funktion_von_systemelement->>A] AND
                  Y:fehler[verhindert_funktion->>B].
```

Wenn ein `systemelement` X eine Schnittstelle zu einem `systemelement` A hat und ein `fehler` Y die Funktionsfähigkeit von `systemelement` A verhindert, so wird er aller Voraussicht nach auch die Funktionsfähigkeit von `systemelement` X beeinträchtigen.

Die Regel lässt sich für den Fall anwenden, dass bei Anwendung der Diagnosefunktion in OntoFMEA zu einem Systemelement mögliche Fehlerursachen angezeigt werden sollen. Hierdurch wird dem Benutzer geholfen, mögliche Fehler zu diagnostizieren. In der Umkehrung erlaubt die Regel, dass zu einer Fehlerursache mögliche zugeordnete Systemelemente ermittelt werden.

10 Fazit

10.1 Zusammenfassung

Aus der wissenschaftlichen Problemstellung der Arbeit ergibt sich das Vorhaben zu zeigen, dass eine simultane Berücksichtigung der beiden generischen Wettbewerbsstrategien Porter´s (Kostenführerschaft und Differenzierung[945]) durch die Integration von Instrumenten des Qualitäts- und Wissensmanagements möglich ist.

Hierzu wird in der vorliegenden Arbeit das Forschungsgebiet der Entwicklung von Ontologien unter Berücksichtigung von Vorgehensmodellen (als Instrumente des Wissensmanagements) auf die Unterstützung der Durchführung einer FMEA (als Instrument des präventiven Qualitätsmanagements) entfaltet.

Es wird ein integrierter Ansatz (OntoFMEA) vorgestellt, der die ontologiebasierte FMEA-Durchführung ermöglicht. Der Ansatz enthält ein OntoFMEA-Vorgehensmodell, das die Entwicklung einer FMEA-Ontologie für ein ontologiebasiertes FMEA-System ermöglicht. Mit Hilfe des OntoFMEA-Vorgehensmodells wird exemplarisch eine FMEA-Ontologie entwickelt, die das Wissen eines FMEA-Formblatts als Struktur repräsentiert. Zudem wird ein Prototyp als weiterer Bestandteil des integrierten Ansatzes vorgestellt, der die entwickelte FMEA-Ontologie nutzt, um eine FMEA-Durchführung mittels eines Wissensbasierten Systems zu gestatten.

Die drei Hauptelemente des integrierten Ansatzes (Vorgehensmodell, FMEA-Ontologie und Prototyp) werden in Hinblick auf ihren Nutzen überprüft und diskutiert. Es wird der Einfluss des integrierten Ansatzes auf die Möglichkeit der Unterstützung einer hybriden Wettbewerbsstrategie aufgezeigt. Zusammengefasst sind in der vorliegenden Arbeit die Ergebnisse entsprechend der Ziele aus Kapitel 1.3 (S. 15):

1. Der *Stand der Forschung* zu den Bereichen FMEA, Ontologien und Vorgehensmodelle wird aufbereitet. Zu Beginn der Untersuchung wurden die drei Instrumente FMEA, Ontologien und Vorgehensmodelle eingehend analysiert. So wurden die Instrumente hinsichtlich ihrer Grundlagen, der Repräsentation und Explikation von Wissen untersucht. Es wurden jeweils beispielhafte Anwendungsfälle genannt. Mit diesen Arbeiten wird die Entwicklung der drei Hauptergebnisse vorbereitet.

2. Es wird ein *integrierter Ansatz (konzeptioneller Rahmen)* zur Nutzen stiftenden Verbindung von FMEA, Ontologien und Vorgehensmodellen entwickelt. Dieser integrierte Ansatz lässt sich in die bereits erwähnten drei Hauptergebnisse gliedern:[946]

945) Genau genommen wird der Spezialfall der Qualitätsführerschaft untersucht. Siehe hierzu auch Kapitel 1, S. 1 ff.

946) Im vorliegenden Sinne lässt sich dieser integrierte Ansatz entsprechend Kapitel 1.3 als konzeptioneller Rahmen auffassen, der die aus Kapitel 1.3 intendierten Ergebnisse 3 bis 5 als teilautonome Objekte umfasst.

I. **OntoFMEA-Vorgehensmodell zur Entwicklung von FMEA-Ontologien**
Das *OntoFMEA-Vorgehensmodell* ermöglicht die Entwicklung von FMEA-Ontologien. Das OntoFMEA-Vorgehensmodell gliedert sich in fünf Hauptphasen (Anforderungsspezifizierung, Wissensakquisition, Konzeptualisierung, Implementierung und Evaluation) sowie die phasenübergreifenden Unterstützungsleistungen Projektmanagement und Dokumentation. Für das OntoFMEA-Vorgehensmodell wurde darauf geachtet, dass es einfach nachzuvollziehen ist und mögliche Werkzeuge nennt, die eine FMEA-Ontologieentwicklung unterstützen. Besondere Berücksichtigung hierbei fanden die FMEA-Formblätter des VDA, anhand derer eine FMEA zumeist durchgeführt wird. Um das Onto-FMEA-Vorgehensmodell zu entwickeln, wurden zu Anfang die Grundsätze von Vorgehensmodellen, wie sie sich in der Literatur finden, genannt. Anschließend wurden Vorgehensmodelle des Software Engineering (sequentieller Software-Life-Cycle-Ansatz, Wasserfall-Ansatz, prototypbasierter Software-Life-Cycle-Ansatz und Spiral-Ansatz), des Knowledge Engineering (Prototyp-Ansatz und Modellbasierter Ansatz) und des Ontology Engineering (IDEF5-Ansatz, Enterprise-Model-Ansatz, TOVE-Ansatz, METHONTOLOGY-Ansatz, On-To-Knowledge-Ansatz und Kollaborativer Ansatz; zusätzlich: MENELAS-Ansatz, KACTUS-Ansatz, SENSUS-Ansatz, ONIONS-Ansatz und OntoClean-Ansatz) analysiert, weil sie von besonderer Bedeutung für das OntoFMEA-Vorgehensmodell sind. Insbesondere bei der Analyse der Vorgehensmodelle des Ontology Engineering wurden Bewertungskriterien entwickelt, die schließlich genutzt wurden, um das vorgestellte OntoFMEA-Vorgehensmodell zu bewerten. Zur semi-formalen Darstellung des OntoFMEA-Vorgehensmodells wurde die Sprache UML (hier: Aktivitätsdiagramme) hinzugezogen.

II. **FMEA-Ontologie mit exemplarischer Wissensbasis der Karl Schumacher Maschinenbau GmbH**
Mit Hilfe des OntoFMEA-Vorgehensmodells lässt sich das Wissen aus den FMEA-Formlättern in einer Wissensbasis strukturiert hinterlegen, indem eine *FMEA-Ontologie* entwickelt wird. Die FMEA-Ontologie mit ihrer exemplarischen Wissensbasis beinhaltet die Hauptkonzepte einer FMEA und die dazugehörigen Relationen und Regeln. Die Ontologie ermöglicht eine Wiederverwendung des hinterlegten Wissens. Für die FMEA-Ontologie wurden Sprachen zur Darstellung vorgestellt und untersucht. In der Arbeit wurde sich aufgrund einer durchgeführten Untersuchung auf die Sprache F-Logic zur Darstellung festgelegt. Unter Rückgriff auf ein vierstufiges Ebenen-Konzept wurden die Bestandteile der FMEA-Ontologie in F-Logic entwickelt. Anhand von Fallbeispielen der Karl Schumacher Maschinenbau GmbH wurde exemplarisch eine Wissensbasis zur Ontologie entwickelt.

III. **Prototypisches Wissensbasiertes System OntoFMEA**
Der vorgestellte *OntoFMEA-Prototyp* des integrierten Ansatzes verwendet die FMEA-Ontologie und ihre Wissensbasis. Der Prototyp unterstützt die Wiederverwendung von Wissen. Es handelt sich bei dem Prototypen um ein webbasiertes System, das mittels JavaServerPages (JSP) auf einem Tomcat-Server die Inferenzmaschine Ontobroker anspricht und die Ergebnisse über einen Internetbrowser dem Benutzer zur Verfügung stellt. Der Prototyp erlaubt die vollständige Durchführung einer FMEA gemäß VDA. Dabei werden Vorschläge zu neu anzulegenden Konzepten, wie etwa Systemelementen, Funktionen oder Fehlern, aus der Wissensbasis abgeleitet und dem Benutzer zur Verfügung gestellt. Des Weiteren erlaubt das System die Durchforstung der Wissensbasis sowie ein erstes einfaches Diagnostizieren von möglichen Gründen für aufgetretene Fehler in Systemen, die bereits in der Wissensbasis (in ähnlicher Form) vorliegen.

Schließlich wird der Nachweis, dass eine simultane hybride Wettbewerbsstrategie verfolgt werden kann, erbracht. Es wird gezeigt, wie die drei Instrumente zusammen als ein integriertes Instrument erweitert auf die Qualität wirken. Hiermit wird des Weiteren gezeigt, wie der integrierte Ansatz (Verbindung von FMEA, Ontologien und Vorgehensmodellen) im instrumentellen Umfeld das wettbewerbliche Umfeld beeinflusst: die Umsetzung einer simultanen *hybriden Wettbewerbsstrategie* wird unterstützt.

10.2 Zukünftige Forschungsansätze

Anhand der Hauptergebnisse Onto-FMEA-Vorgehensmodell, FMEA-Ontologie und Onto-FMEA-Prototyp des integrierten Ansatzes ergeben sich zukünftige Forschungsansätze.

10.2.1 Bereich OntoFMEA-Vorgehensmodell

Das entwickelte OntoFMEA-Vorgehensmodell muss in der vorliegenden Form erst noch den Anforderungen eines „betrieblichen Alltags“ gerecht werden. Es fehlt insofern noch an Erfahrungswerten, die eine abschließende vollständige Evaluation (auch in Bezug auf die Anwendungsbezogenheit[947]) erlauben. Neben der Überprüfung hinsichtlich einer Praxistauglichkeit des OntoFMEA-Vorgehensmodells ergeben sich für die einzelnen Phasen Verbesserungsmöglichkeiten grundsätzlich bei der Erhöhung des Detaillierungsgrads:

- Die Aktivitäten der *Anforderungspezifizierung* zur Identifizierung des Umfelds lassen sich in ein technisches und organisationales Umfeld unterscheiden, die jeweils spezifischen Bedürfnissen gerecht werden müssen. Die Benutzeranforderungen können in Form von Kompetenzfragen formalisiert werden, so dass später eine Evaluation der Ontologie anhand der formalisierten Kompetenzfragen leicht möglich wird.

- In der *Wissensakquisition* können die Iterationen zwischen den Aktionen explizit dargestellt werden.

- In der *Konzeptualisierung* werden insbesondere für die Formulierung von Elementen der Taxonomien und Regeln konkrete Akquisitionsmethoden und Beispiele benötigt, um die Anwendbarkeit des Vorgehensmodells zu erhöhen.

- Die Phase der *Implementierung* kann mehr Beispiele für das Vorgehen zum Transfer einer Konzeptualisierung in eine formale Darstellung enthalten, hierbei sollte auf typische Problemfelder aus anderen Projekten eingegangen werden.

- Während der *Evaluation* kann eine gemeinsame Überprüfung der FMEA-Ontologie durch Benutzer und Entwickler vorgesehen werden. Hierzu müssten weitere Aktionen im Modell Berücksichtigung finden. Mögliche Software-Werkzeuge zur Evaluation sollten angeboten werden.

- Die *phasenübergreifenden Unterstützungsleistungen* – in erster Linie das Projektmanagement – sollten für größere Projekte, die einen erhöhten Planungs-, Durchführungs- und Kontrollaufwand mit sich bringen, weiter ausgearbeitet werden.

Ferner ergeben sich für das OntoFMEA-Vorgehensmodell Änderungen, wenn die beiden anderen Instrumente in ihrem Einsatzbereich erweitert werden. Wenn bspw. beim Onto-FMEA-Prototyp eine Werkzeugintegration (QFD und FMEA) angestrebt wird, wäre es wünschenswert, dass das Vorgehensmodell diese Werkzeugintegration berücksichtigt. In der nächsten

947) Siehe hierzu Kapitel 5.3.3.2.3, S. 171 f.

Evolutionsstufe des OntoFMEA-Vorgehensmodells könnte so ein Vorgehensmodell zur Entwicklung eines ontologiebasierten Qualitätsmanagementsystems angebahnt werden.

10.2.2 Bereich FMEA-Ontologie

Verbesserungsansätze des Prototyps beziehen die FMEA-Ontologie als Verbesserungsmöglichkeit ein. Sollte bspw. angestrebt werden, den Prototyp hinsichtlich weiterer Instrumente des Qualitätsmanagements zu erweitern, so kann es notwendig werden, verschiedene Ontologien zu vereinen. Die Erkenntnisse zur Vereinigung von Ontologien müssen weiter vorangetrieben werden, um unterschiedliche Wissensbasen gemeinsam wiederverwenden zu können. Die einschlägige Literatur hat hierzu bisher tendenziell eher unbefriedigende Lösungen hervorgebracht.[948]

Zusätzlich bieten sich für die vorliegende FMEA-Ontologie folgende spezifischere Aspekte für weitere Forschungsleistungen an:

- Erweiterung der Regelbasis, um präzisere Vorschläge für Instanzen bei der Durchführung einer FMEA unterbreiten zu können. Auch sollte mit Blick auf sich gegenseitig beeinflussende Fehlerursachen eine FMEA-Durchführung ermöglicht werden.
- Erweiterung der Taxonomien zu Systemelementen, Funktionen und Maßnahmen, um wiederum präzisere Vorschläge bei der Durchführung einer FMEA unterbreiten zu können.
- Sofern eine Sprache für die Darstellung von Regeln im Semantic Web als einheitlicher Standard zur Verfügung steht, könnte eine konsequentere Ausrichtung des OntoFMEA-Prototyps auf ein webbasiertes System erreicht werden, indem die FMEA-Ontologie in diese Sprache, die möglicherweise verwandt sein wird zu OWL[949], transferiert wird.

10.2.3 Bereich OntoFMEA-Prototyp

Die Größe der bereits vorliegenden Wissensbasis wirkt sich negativ auf die Antwortzeiten des Softwaresystems aus. So beläuft sich die Rechenzeit für die Ermittlung der Darstellung der Systemelementtaxonomie auf einem Computer mit Intel Pentium 1,7 Gigahertz Prozessor und 512 Mb-Ram bereits auf 3 Min. Dieses Performanzproblem muss für den zukünftigen Einsatz in der betrieblichen Praxis überwunden werden.

Die methodische Unterstützung des Softwaresystems OntoFMEA in Form von Hilfedokumenten und Anleitungen für dessen Bedienung muss erst noch entwickelt werden. Darüber hinaus ist eine intuitiv nachvollziehbare Benutzerführung innerhalb des Programmablaufs zu berücksichtigen, d. h., insbesondere die Benutzerschnittstelle muss zukünftig weiter an die

948) Vgl. Grüninger, Lee (2002), S. 40 f.; Hesse (2002), S. 479; Kalfoglou, Schorlemmer (2003), S. 1 ff.; Noy (2004), S. 382.

949) Siehe hierzu Kapitel 4.2.2.2.4, S. 108 ff.

Bedürfnisse eines Benutzers angepasst werden.[950] Auch eine tiefer gehende Integration des Vorgehensmodells in das Softwaresystem OntoFMEA wäre wünschenswert, bspw. um dem Anwender im oben genannten Sinne (Hilfedokumente) eine erweiterte Hilfestellung bieten zu können.

Insgesamt lässt sich der Prototyp weiterentwickeln in die Richtungen:

- Integration weiterer Instrumente des Qualitätsmanagements zu einem „echten“ Computer Integrated Manufacturing System (CIM),
- Verbesserung der Praxistauglichkeit (z. B. Benutzerführung, Druckfunktion),
- Einbindung weiterer Anwendungsfelder für die Nutzung der Wissensbasis und der Ontologie (wie z. B. Diagnosefunktion, automatisierte Erstellung von Handbüchern oder technischen Spezifikationen)[951] und
- Performanz.

Des Weiteren lässt sich der Prototyp hinsichtlich des berücksichtigten Ablaufs zur Durchführung und des eigentlichen Kerngedankens „FMEA“ verbessern. So wird oftmals bemängelt, dass die FMEA nicht in der Lage ist, Ausfallkombinationen zu berücksichtigen.[952] Stattdessen wird jeder Fehler hinsichtlich seiner Wirkung isoliert betrachtet. Dieses Vorgehen erfolgt zumeist aufgrund einer angestrebten Komplexitätsreduzierung. Für einen Mitarbeiter in einem Unternehmen stellt das Wissen über solche vernetzten Zusammenhänge von sich gegenseitig bedingenden Fehlern ein nicht überschaubares Problem dar. Mit Hilfe von Regeln ließe sich möglicherweise eine „ganzheitlichere“ Sicht auf die Fehlfunktionsstruktur eines Systems erreichen.

---

950) Hierzu bedarf es zunächst zusätzlicher Betrachtungen hinsichtlich möglicher Zielgruppen von Benutzern.

951) Die Diagnosefunktion ist schon rudimentär enthalten, bedarf jedoch einer weiteren Ausarbeitung, um sie als eigenständiges Anwendungsfeld berücksichtigen zu können. Die automatisierte Erstellung von Handbüchern folgt dem Grundgedanken, dass die Funktionalitäten von Systemelementen bereits als Wissen vorliegen und dieses Wissen automatisch zu einem Handbuch über das Gesamtsystem verschriftlicht werden kann.

952) Vgl. Nedeß, Nickel (1992), S. 291, Schloske (1999), S. 21. Siehe hierzu auch die Ausführungen in Kapitel 3.1.3.2, S. 42 ff.

10.3 Schlussbemerkung

Der Verfasser hofft, gezeigt zu haben, wie drei Instrumente des Qualitäts- und Wissensmanagements als integrierter Ansatz im wettbewerblichen Umfeld ihre Wirkung entfalten können. Auch wenn noch weitere Entwicklungsschritte zu unternehmen sind, wurde der Nutzen des vorgestellten Ansatzes deutlich gemacht. Die Verbindung von Ontologien, FMEA und Vorgehensmodellen ist für die Zukunft des Qualitätsmanagements und der Wirkung von Qualität als Erfolgsgröße vielversprechend.

Literaturverzeichnis

Adam (1998)
Adam, D.: Produktions-Management, 9. Aufl., Wiesbaden 1998.

Alan (2002)
Alan, Y.: Anforderungen an den KOWIEN-Prototypen. Projektbericht 6/2002, Projekt KOWIEN, Institut für Produktion und Industrielles Informationsmanagement, Universität Essen, Essen 2002.

Alan, Alparslan et al. (2003)
Alan, Y.; Alparslan, A.; Dittmann, L.: Werkzeuge zur Sicherstellung der Adaptibilität des KOWIEN-Vorgehensmodells. Projektbericht 6/2003, Projekt KOWIEN, Institut für Produktion und Industrielles Informationsmanagement, Universität Duisburg-Essen (Campus Essen), Essen 2003.

Albert (1987)
Albert, H.: Kritik der reinen Erkenntnislehre, Tübingen 1987.

Albert (1991)
Albert, H.: Traktat über kritische Vernunft, 5. Aufl., Tübingen 1991.

Algedri, Frieling et al. (2000)
Algedri, J.; Frieling, E.; Katschke, M.; Barchfeld, K.-H.: Optimierung mit System – Human-FMEA deckt Zusammenhänge zwischen Produkt- und menschlichen Handlungsfehlern auf. In: QZ – Qualität und Zuverlässigkeit, Jg. 45 (2000), Nr. 1, S. 68-71.

Alparslan, Dittmann et al. (2002)
Alparslan, A.; Dittmann, L.; Zelewski, S.; Ilgen, A.: Wissensmanagement im Anlagenbau – Computergestütztes Management von Wissen über Mitarbeiterkompetenzen. In: Industrie Management, Jg. 18 (2002), Nr. 6, S. 45-48.

Angele, Fensel et al. (1993)
Angele, J.; Fensel, D.; Landes, D.; Neubert, S.; Studer, R.: Model-Based and Incremental Knowledge Engineering: The MIKE Approach. In: Cuena, J. (Hrsg.): Knowledge Oriented Software Design. IFIP-Transactions A-27, Amsterdam 1993, o. S.
URL: http://www.ontoprise.de/members/angele/pubs/mike.pdf, Zugriff am 17.11.2004, S. 1-29.

Angele, Fensel et al. (1995)
Angele, J.; Fensel, D.; Landes, D.; Neubert, S.: Modellbasiertes und Inkrementelles Knowledge Engineering: der MIKE-Ansatz. In: KI – Künstliche Intelligenz, Jg. (1995), Nr. 1, S. 16-21.

Angele, Fensel et al. (1998)
Angele, J.; Fensel, D.; Studer, R.: Vorgehensmodelle für die Entwicklung wissensbasierter Systeme. In: Kneuper, R.; Müller-Luschnat, G.; Oberweis, A. (Hrsg.): Vorgehensmodelle für die betriebliche Anwendungsentwicklung, Stuttgart 1998, S. 168-188.

Angele, Lausen (2004)
Angele, J.; Lausen, G.: Ontologies in F-Logic. In: Staab, S.; Studer, R. (Hrsg.): Handbook on Ontologies, Berlin 2004, S. 29-50.

Angele, Sure (2002)
Angele, J.; Sure, Y.: EFFORT – Evaluation Framework For Ontologies and Related Technologies. Technical Report, Institut AIFB, Universität Karlsruhe und Ontoprise GmbH, Karlsruhe 2002.
URL: http://209.182.10.77/public_ontoprise.html, Zugriff am 9.02.2005.

Antoniou, van Harmelen (2004)
Antoniou, G.; van Harmelen, F.: Web Ontology Language: OWL. In: Staab, S.; Studer, R. (Hrsg.): Handbook on Ontologies, Berlin 2004, S. 67-92.

Apke, Bremer et al. (2004)
Apke, S.; Bremer, A.; Dittmann, L.: Konstruktion einer Kompetenz-Ontologie, dargestellt am Beispiel der Deutschen Montan Technologie GmbH (DMT). Projektbericht 6/2004, Projekt KOWIEN, Institut für Produktion und Industrielles Informationsmanagement, Universität Duisburg-Essen (Campus Essen), Essen 2004.

Apke, Dittmann (2003a)
Apke, S.; Dittmann, L.: Analyse von Vorgehensmodellen aus dem Software, Knowledge und Ontologies Engineering. Projektbericht 1/2003, Projekt KOWIEN, Institut für Produktion und Industrielles Informationsmanagement, Universität Duisburg-Essen (Campus Essen), Essen 2003.

Apke, Dittmann (2003b)
Apke, S.; Dittmann, L.: Generisches Vorgehensmodell KOWIEN Version 1.0. Projektbericht 4/2003, Projekt KOWIEN, Institut für Produktion und Industrielles Informationsmanagement, Universität Duisburg-Essen (Campus Essen), Essen 2003.

Apke, Dittmann (2004)
Apke, S.; Dittmann, L.: Generisches Vorgehensmodell KOWIEN Version 2.0. Projektbericht 7/2004, Projekt KOWIEN, Institut für Produktion und Industrielles Informationsmanagement, Universität Duisburg-Essen (Campus Essen), Essen 2004.

Arpírez, Corcho et al. (2001)
Arpírez, J.; Corcho, O.; Fernández López, M.; Gómez-Pérez, A.: WebODE: a Scalable Workbench for Ontological Engineering. In: Gil, Y.; Musen, M.; Shavlik, J. (Hrsg.): Proceedings of the International Conference on Knowledge Capture – K-CAP-01, Victoria 2001, S. 6-13.

Baader, Calvanese et al. (2003)
Baader, f.; Calvanese, D.; McGuinness, D.; Nardi, D.; Patel-Schneider, P. (Hrsg.): The Description Logic Handbook – Theory, Implementation and Applications, Cambridge 2003.

Baader, Horrocks et al. (2004)
Baader, F.; Horrocks, I.; Sattler, U.: Description Logics. In: Staab, S.; Studer, R. (Hrsg.): Handbook on Ontologies, Berlin 2004, S. 3-28.

Baclawski, Kokar et al. (2002)
Baclawski, K.; Kokar, M. K; Kogut, P. A.; Hart, L.; Smith, J.; Letkowski, J.; Emery, P.: Extending the Unified Modeling Language for ontology development. In: Software Systems Modelling, Jg. 1 (2002), Nr. 2, S. 142-156.

Bahr (1999)
Bahr, M.: Kundenzufriedenheit als Strategieelement in der Bauindustrie, Berlin 1999.

Bailin, Truszkowski (2002)
Bailin, S.C.; Truszkowski, W.: Ontology negotiation between intelligent information agents. In: The Knowledge Engineering Review, Jg. 17 (2002), Nr. 1, S. 7-19.

Bamberger, Wrona (2004)
Bamberger, I.; Wrona, T.: Strategische Unternehmensführung, München 2004.

Bartlett, Ghoshal (2002)
Bartlett, C. A.; Ghoshal, S.: Building Competitive Advantage Through People. In: MIT Sloan Management Review, Jg. 43 (2002), Nr. 2, S. 34-41.

Baskerville, Pries-Heje (1999)
Baskerville, R.; Pries-Heje, J.: Managing Knowledge Capability and Maturity. In: Larsen, T.; Levine, L.; DeGross, J.: Information Systems: Current Issues and Future Changes, Laxenburg 1999, S. 175-196.

Bateman, Kasper et al. (1989)
Bateman, J.; Kasper, R.; Moore, J.; Whitney, R.: A general organization of knowledge for natural language processing: The Penman Upper Model. Technical Report Information Sciences Institute, Marina del Rey 1989.

Bateman, Magnini et al. (1994)
Bateman, J.; Magnini, B.; Rinaldi, F.: The Generalized {Italian, German, English} Upper Model. In: Mars, N. (Hrsg.): Comparison of Implemented Ontologies, Proceedings of the ECAI'94 Workshop, Amsterdam 1994, S. 35-45.

Baumgarten (1990)
Baumgarten, B.: Petri-Netze: Grundlagen und Anwendungen, Mannheim 1990.

Beckermann (1997)
Beckermann, A.: Einführung in die Logik, Berlin 1997.

Beierle, Kern-Isberner (2003)
Beierle, C.; Kern-Isberner, G.: Methoden wissensbasierter Systeme – Grundlagen, Algorithmen, Anwendungen, 2. Aufl., Wiesbaden 2003.

Benjamins, Fensel et al. (1998)
Benjamins, V. R.; Fensel, D.; Gómez-Peréz, A.: Knowledge Management through Ontologies. In: Reimer, U.: Proceedings of the 2nd Conference on Practical Aspects of Knowledge Management (PAKM98), Aachen 1998, S. 5.1-5.11.

Berger (1994)
Berger, H.: Ansätze des Qualitätsmanagements für eine zukunftsweisende Produktentwicklung, Dissertation Universität St. Gallen, St. Gallen 1994.

Berners-Lee, Hendler et al. (2001)
Berners-Lee, T.; Hendler, J.; Lassila, O.: The Semantic Web. In: Scientific American, Jg. 284 (2001), Nr. 5, S. 34-43.

Berthold, Krämer et al. (1995)
Berthold, B.; Krämer, A.; Peter, G.; Wirth, R.: Wissensbasierte Systemanalyse. In: QZ – Qualität und Zuverlässigkeit, Jg. 40 (1995), Nr. 6, S. 714-718.

Bertin, Buciol et al. (1998)
Bertin, A.; Buciol, F.; Stefanini, A.: Towards Industrial Application of Intelligent Training Systems. In: Expert Systems, Jg. 15 (1998), Nr. 1, S. 1-21.

Bibel (1993)
Bibel, W.: Wissensrepräsentation und Inferenz – Eine grundlegende Einführung. In Zusammenarbeit mit Hölldobler, S. und Schaub, T., Braunschweig 1993.

Bickenbach, Freyler at al. (2000)
Bickenbach, D.; Freyler, A.; Ringwald, U.; Schramm, K.: Wissensmanagement und Qualitätssicherung in der Organisationsentwicklung. In: Organisationsberatung – Supervision – Clinical Management, Jg. 7 (2000), Nr. 4, S. 351-366.

Biethahn, Muksch et al. (2000)
Biethahn, J.; Muksch, H.; Ruf, W.: Ganzheitliches Informationsmanagement, Band 1: Grundlagen, 5. Aufl., München 2000.

Biezunski (2003)
Biezunski, M.: Introduction to the Topic Maps Paradigm. In: Park, J. (Hrsg.); Hunting, S. (technischer Hrsg.): XML Topic Maps – Creating and Using Topic Maps for the Web, Boston 2003, S. 17-30.

Birkhofer (1980)
Birkhofer, H.: Analyse und Synthese der Funktionen technischer Produkte, VDI-Fortschritt-Bericht 1-70, Düsseldorf 1980.

Blau (1978)
Blau, U.: Die dreiwertige Logik der Sprache – Ihre Syntax, Semantik und Anwendung in der Sprachanalyse, Berlin 1978.

Boehm (1988)
Boehm, B. W.: A Spiral Model of Software Development and Enhancement. In: IEEE Computer, Jg. 21 (1998), Nr. 3, S. 61-72.

Boehm (1989)
Boehm, B. W.: Verifying and Validating Software Requirements and Design Specifications. In: Boehm, B. W. (Hrsg.): Software Risk Management, Washington 1989, S. 205-218.

Böhm, Wenger (1996)
Böhm, R.; Wenger, S.: Methoden und Techniken der System-Entwicklung, 2. Aufl., Zürich 1996.

Boman, Bubenko et al. (1997)
Boman, M.; Bubenko, J. A.; Johannesson, P.; Wangler, B.: Conceptual Modelling, London 1997.

Booch, Rumbaugh et al. (1999)
Booch, G., Rumbaugh, J.; Jacobson, I.: Das UML-Benutzerhandbuch, Bonn 1999.

Borgida (1996)
Borgida, A.: On the Relative Expressiveness of Description Logics and Predicate Logics. In: Artificial Intelligence, Jg. 82 (1996), Nr. 1-2, S. 353-367.

Borrmann, Komnik (2002)
Borrmann, A.; Komnik, S.; Matèrne, J.; Landgrebe, G.; Räkermann, M.; Sauer, J.: Rational Rose und UML – Anleitung zum Praxiseinsatz, Bonn 2005.

Borst (1997)
Borst, W. N.: Construction of Engineering Ontologies for Knowledge Sharing and Reuse, Dissertation Universität Enschede, Enschede 1997.

Borst, Benjamin et al. (1996)
Borst, P.; Benjamin, J.; Wielinga, B.; Akkermans, H.: An Application of Ontology Construction. In: o. Hrsg.: Workshop on Ontological Engineering, ECAI ´96, Budapest 1996, S. 5-16.

Bouaud, Bachimont et al. (1994)
Bouaud, J.; Bachimont, B.; Charlet, J.; Zweigenbaum, P.: Acquisition and Structuring of an Ontology within Conceptual Graphs. In: o. Hrsg.: Proceedings of ICCS '94, Workshop on Knowledge Acquisition using Conceptual Graph Theory, University of Maryland, College Park 1994, S. 1-25.

Bouaud, Bachimont et al. (1995)
Bouaud, J.; Bachimont, B.; Charlet, J.; Zweigenbaum, P.: Methodological Principles for Structuring an Ontology. In: o. Hrsg.: Workshop on Basic Ontological Issues in Knowledge Sharing (in conjunction with IJCAI-95), Montreal 1995, S. 1-7. Auch erschienen als DIAM Rapport Interne RI-95-148.
URL: http://www.biomath.jussieu.fr, Zugriff am 18.02.2003.

Brachman (1979)
Brachman, R. J.: On the Epistemological Status of Semantic Networks. In: Findler, N. V. (Hrsg.): Associative Networks: Representation and Use of Knowledge by Computers, New York 1979, S. 3-50.
Zitiert gemäß Nachdruck in: Brachman, R. J.; Levesque, H. J. (Hrsg.): Readings in Knowledge Representation, Los Altos 1985, S. 191-215.

Brachman, Levesque (1985)
Brachman, R. J.; Levesque, H. J.: Introduction. In: Brachman, R. J.; Levesque, H. J. (Hrsg.): Readings in Knowledge Representation, Los Altos 1985, S. XIII-XIX.

Bremer (1998)
Bremer, G.: Genealogie von Entwicklungsschemata. In: Kneuper, R.; Müller-Luschnat, G.; Oberweis, A. (Hrsg.): Vorgehensmodelle für die betriebliche Anwendungsentwicklung, Stuttgart 1998, S. 32-59.

Bröhl, Dröschel (1995)
Bröhl, A.-P.; Dröschel, W.: Einführung in das V-Modell. In: Bröhl, A.-P.; Dröschel, W. (Hrsg.): Das V-Modell – Der Standard für die Softwareentwicklung mit Praxisleitfaden, 2. Aufl., München 1995.

Bruhn (1997)
Bruhn, M.: Qualitätsmanagement für Dienstleistungen. 2. Aufl., Berlin 1997.

Bruhn (2000)
Bruhn, M.: Qualitätssicherung im Dienstleistungsmarketing. In: Bruhn, M.; Stauss, B. (Hrsg.): Dienstleistungsqualität – Konzepte, Methoden, Erfahrungen, 3. Aufl., Wiesbaden 2000, S. 21-48.

Bruhn, Georgi (1999)
Bruhn, M.; Georgi, D.: Kosten und Nutzen des Qualitätsmanagements – Ansatzpunkte einer Wirtschaftlichkeitsanalyse des Qualitätsmanagements. In: Die Unternehmung, Jg. 53 (1999), Nr. 3, S. 177-191.

Büchel (2002)
Büchel, G.: Ontologien und eine Aufgabenstellung der Computerlinguistik. In: Willée, G.; Schröder, B.; Schmitz, H.-C. (Hrsg.): Computerlinguistik – Was geht, was kommt?, Sankt Augustin 2002, S. 34-40.

Bullinger, fähnrich (1997)
Bullinger, H.-J.; Fähnrich, K.-P.: Betriebliche Informationssysteme – Grundlagen und Werkzeuge der methodischen Softwareentwicklung, Berlin 1997.

Burkhardt (1997)
Burkhardt, R.: UML – Unified Modeling Language: Objektorientierte Modellierung für die Praxis, Bonn 1997.

Buzzell, Gale (1989)
Buzzell, R. D.; Gale, B. T.: Das PIMS-Programm – Strategien und Unternehmenserfolg, Wiesbaden 1989.

Carlson, McCullen et al. (1996)
Carlson, W. D.; McCullen, L. R.; Miller, G. H.: Why SAE J1739? In: SAE Transactions, Jg. 104 (1996), Nr. 5, S. 481-488.

Carlson, Nirenburg (1990)
Carlson, L.; Nirenburg, S.: World Modeling for NLP, Technical Report CMU-CMT-90-121, Pittsburgh 1990.

Chandrasekaran, Josephson et al. (1999)
Chandrasekaran, B.; Josephson, J. R.; Benjamins V. R.: What are Ontologies, and Why Do We Need Them? In: IEEE Intelligent Systems, Jg. 14 (1999), Nr. 1, S. 20-26.

Chaudhri, Farquhar et al. (1998)
Chaudhri, V. K.; Farquhar, A.; Fikes, R.; Karp, P. D.; Rice, J. P.: OKBC: a programmatic foundation for knowledge base interoperability. In AAAI Press (Hrsg.): AAAI '98/IAAI '98: Proceedings of the 15th National/10th Conference on Artificial Intelligence/Innovative Applications of Artificial Intelligence, Madison 1998, S. 600-607.

Chen (1976)
Chen, P. P.: The Entity-Relationship Model – Toward a Unified View of Data. In: ACM Transactions on Database Systems, Jg. 1 (1976), Nr. 1, S. 9-36.

Clark, Porter (1999)
Clark, P.; Porter, B.: KM – The Knowledge Machine User Manual, Universität Austin (Texas), Austin 1999.

Codex Alimentarius (1999)
Codex Alimentarius Commission: Recommended International Code of Practice – General Principles of Food Hygiene (CAC/RCP 1-1969, Rev. 3 (1997)). In: Codex Alimentarius: General requirements (food hygiene), Ausgabe 1 B, überarbeitete Version, Rom 1999, S. 1-30.

Corcho, Fernández-López et al. (2003)
Corcho, O.; Fernández-López, M.; Gómez-Pérez, A.: Methodologies, tools and languages for building ontologies. Where is the meeting point? In: Data & Knowledge Engineering, Jg. 46 (2003), Nr. 1, S. 41-64.

Corcho, Gómez-Pérez (2000)
Corcho, O.; Gómez-Pérez, A.: Evaluating Knowledge Representation and Reasoning Capabilities of Ontology Specification Languages. In: Benjamins, V. R.; Gómez-Pérez, A.; Guarino, N.; Uschold, M. (Hrsg.): Workshop on Applications of Ontologies and Problem Solving Methods, 14th European Conference on Artificial Intelligence (ECAI ´00), Berlin 2000, S. 3.1-3.9.

Corcho, Gómez-Pérez (2004)
Corcho, O.; Gómez-Pérez, A.: Ontology Translation Approaches for Interoperability: A Case Study with Protégé-2000 and WebOde. In: Motta, E.; Shadbolt, N.; Stutt, A.; Gibbins, N. (Hrsg.): Engineering Knowledge in the Age of the Semantic Web, Proceedings of the 14th International Conference on Knowledge Engineering and Knowledge Management (EKAW 2004), Lecture Notes in Artificial Intelligence, Nr. 3257, Berlin 2004, S. 30-46.

Corsten (1998)
Corsten, H.: Grundlagen der Wettbewerbsstrategie, Stuttgart 1998.

Corsten, Will (1994)
Corsten, H.; Will, T.: Wettbewerbsstrategien und Produktionsorganisation. In: Corsten, H. (Hrsg.): Handbuch Produktionsmanagement – Strategie, Führung, Technologie, Schnittstellen, Wiesbaden 1994, S. 260-273.

Cranefield, Willmott (2002)
Cranefield, S.; Willmott, S.; Finin, T.: Introduction to the special issue on ontologies in agent systems. In: The Knowledge Engineering Review, Jg. 17 (2002), Nr. 1, S. 1-5.

Cycorp (2004)
Cycorp, Inc.: Semantic Knowledge Source Integration, White Paper, Austin 2004.
URL: http://www.cyc.com/cyc/technology/whitepapers_dir/sksi%20_data_sheet.pdf, Zugriff am 18.03.2007.

Decker (1998)
Decker, S.: On Domain-Specific Declarative Knowledge Representation and Database Languages. In: Borgida, A.; Chaudri, V.; Staudt M. (Hrsg.): Proceedings of the 5th Knowledge Representation meets Databases Workshop (KRDB ´98), Seattle 1998, S. 9.1-9.7.

Decker (2002)
Decker, S.: Semantic Web Methods for Knowledge Management, Dissertation Universität Karlsruhe, Karlsruhe 2002.

Deutsch, Mücke (1999)
Deutsch, H.; Mücke, K.: Sicherheit für Einzelanfertigungen – Mittels Sondermaschinen-FMEA Konstruktion und Anlagenverfügbarkeit analysieren. In: QZ – Qualität und Zuverlässigkeit, Jg. 44 (1999), Nr. 7, S. 897-899.

DGQ-Band 11-04 (2002)
DGQ-Band 11-04: Managementsysteme – Begriffe: Ihr Weg zu klarer Kommunikation, Deutsche Gesellschaft für Qualität e.V. (Hrsg.), 7. Aufl., Berlin 2002.

DGQ-Band 13-11 (2001)
DGQ-Band 13-11: FMEA – Fehlermöglichkeits- und Einflussanalyse, Deutsche Gesellschaft für Qualität e.V. (Hrsg.), 2. Aufl., Berlin 2001.

Dietzsch, Althaus et al. (1999)
Dietzsch, M.; Althaus, K.; Brandner, T.: Fehler früh erkennen – Produktentwicklungs-FMEA zur Entwicklung komplexer Produkte. In: QZ – Qualität und Zuverlässigkeit, Jg. 44 (1999), Nr. 11, S. 1394-1398.

Dimitrov, Schmietendorf et al. (2000)
Dimitrov, E.; Schmietendorf, A.; Wipprecht, M.: Effizienter Einsatz eines Vorgehensmodells für die objektorientierte Entwicklung. In: Andelfinger, U.; Herzwurm, G.; Mellis, W.; Müller-Luschnat, G. (Hrsg.): Vorgehensmodelle: Wirtschaftlichkeit, Werkzeugunterstützung und Wissensmanagement, 7. Workshop der Fachgruppe 5.11 der Gesellschaft für Informatik e.V., Aachen 2000, S. 53-66.

DIN 1463-1:1987
Deutsches Institut für Normung e.V. (Hrsg.): DIN 1463 Teil 1, Erstellung und Weiterentwicklung von Thesauri, Berlin 1987.

DIN 25419:1985
Deutsches Institut für Normung e.V. (Hrsg.): DIN 25419, Ereignisablaufanalyse – Verfahren, graphische Symbole und Auswertung, Berlin 1985.

DIN 25424-1:1981
Deutsches Institut für Normung e.V. (Hrsg.): DIN 25424, Teil 1, Fehlerbaumanalyse – Methode und Bildzeichen, Berlin 1981.

DIN 25448:1990
Deutsches Institut für Normung e.V. (Hrsg.): DIN 25448, Ausfalleffektanalyse (Fehler-Möglichkeits- und Einflußanalyse), Berlin 1990.

DIN 40150:1979
Deutsches Institut für Normung e.V. (Hrsg.): DIN 40150, Begriffe zur Ordnung von Funktions- und Baueinheiten, Berlin 1979.

DIN E 2342:2004
Deutsches Institut für Normung e.V. (Hrsg.): DIN 2342, Begriffe der Terminologielehre, Entwurf, vorgesehen als Ersatz für DIN 2342-1:1992, Berlin 2004.

DIN EN ISO 8402:1995
Deutsches Institut für Normung e.V. (Hrsg.): DIN 8402, Qualitätsmanagement Begriffe, Berlin 1995.

DIN EN ISO 9000:2000
DIN EN ISO 9000:2000. In: Deutsches Institut für Normung e.V. (Hrsg.): Qualitätsmanagement und Statistik – Begriffe: Normen, DIN-Taschenbuch 223, 3. Aufl., Berlin 2001, S. 116-177.

DIN EN ISO 9001:2000
DIN EN ISO 9001:2000. In: Deutsches Institut für Normung e.V. (Hrsg.): Qualitätsmanagement: Normen, DIN-Taschenbuch 226, 3. Aufl., Berlin 2001, S. 1-66.

DIN EN ISO 9004:2000
DIN EN ISO 9004:2000. In: Deutsches Institut für Normung e.V. (Hrsg.): Qualitätsmanagement: Normen, DIN-Taschenbuch 226, 3. Aufl., Berlin 2001, S. 67-164 .

Dittmann, Penzel (2004)
Dittmann, L.; Penzel, J.: Platons Gütekriterium für Ontologien. In: Frank, U. (Hrsg.): Wissenschaftstheorie in Ökonomie und Wirtschaftsinformatik, Wiesbaden 2004, S. 457-478.

Dittmann, Rademacher (2004)
Dittmann, L.; Rademacher, T.: OntoFMEA: ein Softwaretool für ontologiebasierte Fehlermöglichkeits- und -einflussanalysen – konzeptionelle Entwicklung und prototypische Implementierung. Projektbericht 11/2004, Projekt KOWIEN, Institut für Produktion und Industrielles Informationsmanagement, Universität Duisburg-Essen (Campus Essen), Essen 2004.

Dittmann, Rademacher et al. (2004)
Dittmann, L.; Rademacher, T.; Zelewski, S.: Combining Knowledge Management and Quality Management Systems. In: o. Hrsg.: 48th Congress of European Organization for Quality (EOQ), Congress Materials, Moskau 2004, o. S.

Dittmann, Schütte et al. (2003)
Dittmann, L.; Schütte, R.; Zelewski, S.: Darstellende Untersuchung philosophischer Probleme mit Ontologien. In: Freyberg, K.; Petsche, H.-J.; Klein, B. (Hrsg.): Proceedings of the WM 2003 Workshop on Knowledge Management and Philosophy, CEUR Workshop Proceedings, Vol. 85, Luzern 2003. S. 3.1-3.13.

Dittmann, Zelewski (2004)
Dittmann, L.; Zelewski, S.: Integrating Computer-based Systems of Knowledge and Quality Engineering to Manage Skills. In: Weckenmann, A. (Hrsg.): 8th International Symposium on Measurement and Quality Control in Production (ISMQC 2004), VDI-Berichte Band Nr. 1860, Erlangen 2004, S. 181-187.

Dobry (2003)
Dobry, A.: Global denken, lokal handeln – FMEA: Effektiver Umgang mit komplexen Systemen. In: QZ – Qualität und Zuverlässigkeit, Jg. 48 (2003), Nr. 11, S. 1096-1097.

Durkin (1998)
Durkin, J.: Expert System Development Tools. In: Liebowitz, J. (Hrsg.): The Handbook of Applied Expert Systems, Boca Raton 1998, S. 4.1-4.26.

Eberhardt (2003)
Eberhardt, O.: Gefährdungsanalyse mit FMEA: die Fehler-Möglichkeits- und Einflussanalyse gemäß VDA-Richtlinie, Renningen 2003.

Eckstein (2000)
Eckstein, R.: XML: kurz & gut, 2. korrigierter Nachdruck, Köln 2000.

Edenhofer, Baumann et al. (2002)
Edenhofer, B.; Baumann, T.; Esch, F.-J.; Knoop, O.: Richtig smart erst gemeinsam – System-FMEA: Risikoabsicherung in Fahrzeugprojekten nur durch enge Einbindung aller Partner. In: QZ – Qualität und Zuverlässigkeit, Jg. 47 (2002), Nr. 7, S. 732-735.

EG-Richtlinie 43 (1993)
Richtlinie 93/43/EWG des Rates vom 14. Juni 1993 über Lebensmittelhygiene, Amtsblatt Nr. L 176 vom 20.07.1993, S. 29.

Epstein (1994)
Epstein, R. L.: The Semantic Foundations of Logic – Predicate Logic, New York 1994.

Erdmann (2001)
Erdmann, M.: Ontologien für die konzeptuelle Modellierung der Semantik von XML. Dissertation Universität Karlsruhe, Karlsruhe 2001.

Eriksson, Penker (2000)
Eriksson, H.-E.; Penker, M.: Business modeling with UML: business patterns at work, New York 2000.

Eurich (1988)
Eurich, C.: Die Megamaschine – Vom Sturm der Technik auf das Leben und Möglichkeiten des Widerstands, Darmstadt 1988.

ExBa (2003)
forum! GmbH marketing + communications; DGQ – Deutsche Gesellschaft für Qualität e. V. (Hrsg.): ExBa 2003 – Benchmarkstudie zur Excellence in der deutschen Wirtschaft, Mainz 2003.

Fachinformationszentrum Technik (2000)
Fachinformationszentrum Technik e.V.: Thesaurus Technik und Management, 2. Aufl., Frankfurt/Main 2000.

Farmer, Vlk (2005)
Farmer, K.; Vlk, T.: Internationale Ökonomie – Eine Einführung in die Theorie und Empirie der Weltwirtschaft, Wien 2005.

Farquhar, Fikes et al. (1996)
Farquhar, A.; Fikes, R.; Rice, J.: The Ontolingua Server: a Tool for Collaborative Ontology Construction. In: Proceedings of the 10th Knowledge Acquisition for Knowledge-Based Systems Workshop (1996), Banff 1996, S. 44.1-44.19.

Farrar, Bateman (2004)
Farrar, S.; Bateman, J.: General Ontology Baseline, OntoSpace Projektbericht, Bremen 2004.
URL: http://134.102.58.154/documents/techreport/FarrarBateman04-i1-d1.pdf, Zugriff am 18.02.2005.

Feigenbaum (1991)
Feigenbaum, A. V.: Total Quality Control, 3. Aufl., New York 1991.

Fensel (1995)
Fensel, D.: The Knowledge Acquisition and Representation Language KARL, Dissertation Universität Karlsruhe 1993, Berlin 1995.

Fensel (2004)
Fensel, D.: Ontologies: A Silver Bullet for Knowledge Management and Electronic Commerce, 2. Aufl., Berlin 2004.

Fensel, angele et al. (1998)
Fensel, D.; Angele, J.; Studer, R.: The Knowledge Acquisition and Representation Language KARL. In: IEEE Transaction in Knowledge and Data Engineering (TKDE), Jg. 10 (1998), Nr. 4, S. 527-550.

Fensel, Angele et al. (1999)
Fensel, D.; Angele, J.; Decker, S.; Erdmann, M.; Schnurr, H.-P.; Staab, S.; Studer, R.; Witt, A.: On2broker: Semantic-based access to information sources at the WWW. In: de Bra, P.; Leggett, J. (Hrsg.): Proceedings of the World Conference on the WWW and Internet (WebNet99) 1999, S. 366-371.

Fensel, Decker et al. (1998)
Fensel, D.; Decker, S.; Erdmann, M.; Studer, R.: Ontobroker. How to make the WWW Intelligent. Technical Report 376, Institut AIFB, Universität Karlsruhe, Karlsruhe 1998.

Fensel, Motta et al. (1997)
Fensel, D.; Motta, E.; Decker, S.; Zdrahal, Z.: The Use of Ontologies for Specifying Tasks and Problem Solving Methods: A Case Study. In: Plaza, E.; Benjamins, R. (Hrsg.): Proceedings of the 10th European Workshop on Knowledge Acquisition, Modeling, and Management (EKAW'97), Sant Feliu de Guíxols 1997, S. 113-128.

Fernández López (1999)
Fernández López, M.: Overview Of Methodologies For Building Ontologies. In: Benjamins, V.R.; Chandrasekaran, B.; Gomez-Perez, A.; Guarino, N.; Uschold, M. (Hrsg.): Proceedings of the IJCAI-99 Workshop on Ontologies and Problem-Solving Methods, Stockholm 1999, S. 4.1-4.13.

Fernández López, Gómez-Pérez et al. (1999)
Fernández López, M.; Gómez-Pérez, A.; Pazos Sierra, J.; Pazos Sierra, A.: Building a Chemical Ontology Using METHONTOLOGY and the Ontology Design Environment. In: IEEE Intelligent Systems: Special Issue on Uses of Ontologies, Jg. 14 (1999), Nr. 1, S. 37-46.

Fernández, Gómez-Pérez et al. (1997)
Fernández, M.; Gómez-Pérez, A.; Juristo, N.: METHONTOLOGY: From Ontological Art Towards Ontological Engineering. In: Farquar, A.; Grüninger, M. (Hrsg.): Ontological Engineering – Papers from the 1997 AAAI Spring Symposium, Stanford 1997, S. 33-40.

Ferstl, Sinz (1993)
Ferstl, O.; Sinz, E.: Geschäftsprozessmodellierung. In: Wirtschaftsinformatik, Jg. 35 (1993), Nr. 6, S. 589-592.

Ferstl, Sinz (1994)
Ferstl, O.; Sinz, E.: Der Ansatz des semantischen Objektmodells zur Modellierung von Geschäftsprozessen, Bamberger Beiträge, Nr. 23, Bamberg 1994.

Fischer, Biskup et al. (1998)
Fischer, T.; Biskup, H.; Müller-Luschnat, G.: Begriffliche Grundlagen für Vorgehensmodelle. In: Kneuper, R.; Müller-Luschnat, G.; Oberweis, A. (Hrsg.): Vorgehensmodelle für die betriebliche Anwendungsentwicklung, Leipzig 1998, S. 13-31.

Fischer, Grünbacher et al. (1999)
Fischer, C.; Grünbacher, P.; Widmann, M.: Introduction of a Process Model for Object-Oriented, Component-Oriented Software Development at Axioma Information Systems. In: Kneuper, R.; Verlage, M. (Hrsg.): Vorgehensmodelle, Prozessverbesserung und Qualitätsmanagement, 6. Workshop der Fachgruppe 5.1.1 der Gesellschaft für Informatik e.V., Stuttgart 1999, S. 43-50.

Fleck (1995)
Fleck, G.: Hybride Wettbewerbsstrategien – Zur Synthese von Kosten- und Differenzierungsstrategien, Wiesbaden 1995.

Fluit, Sabou et al. (2004)
Fluit, C.; Sabou, M.; van Harmelen, F.: Supporting User Tasks through Visualisation of Light-weight Ontologies. In: Staab, S.; Studer, R. (Hrsg.): Handbook on Ontologies, Berlin 2004, S. 415-432.

Fowler (2004)
Fowler, M.: UML Distilled – A Brief Guide to the Standard Object Modeling Language, 3. Aufl., Boston 2004.

Fox, Grüninger (1997)
Fox, M. S.; Grüninger, M.: On Ontologies and Enterprise Modelling. In: Kosanke, K.; Nell, J. (Hrsg.): Proceedings of the International Conference on Enterprise Integration Modelling Technology, Berlin 1997, S. 190-200.

Fox, Grüninger (1998)
Fox, M. S., Grüninger, M.: Enterprise Modelling. In: AI Magazine, Jg. 19 (1998), Nr. 3, S. 109-121.

FQS (1994)
Forschungsgemeinschaft Qualitätssicherung e.V. (FQS) (Hrsg.): Rechnergestützte, wissensbasierte Erstellung von Fehlermöglichkeits- und Einflussanalysen (FMEA), FQS-Schrift 85-02, Berlin 1994.

Frank (1998)
Frank, U.: Die Evaluation von Artefakten: Eine zentrale Herausforderung der Wirtschaftsinformatik, Koblenz 1998.
URL: http://www.wi-inf.uni-essen.de/FGFrank/documents/Zeitschriftenartikel/LinzEvaluation.pdf, Zugriff am 18.03.2007, S. 1-23.

Frank (2000)
Frank, U.: Unified Modeling Language (UML) – ein bedeutsamer Standard für die konzeptionelle Modellierung. In: Das Wirtschaftsstudium, Jg. 29 (2000), Nr. 5, S. 709-718.

Friedland, Allen (2003)
Friedland, N. S.; Allen, P. G.: The Halo Pilot: Towards A Digital Aristotle, Seattle 2003.
URL: http://www.projecthalo.com/content/docs/halopilot_vulcan_finalreport.pdf, Zugriff am 18.3.2007, S. 1-16.

Gale, Buzzell (1989)
Gale, B. T.; Buzzell, R. D.: Market Perceived Quality: Key Strategic Concept. In: Planning Review, Jg. 17 (1989), Nr. 1, S. 6-15 und 48.

Gangemi, Pisanelli et al. (1998)
Gangemi, A.; Pisanelli, D. M.; Steve, G.: Ontology Integration: Experiences with Medical Terminologies. In: Guarino, N. (Hrsg.): Formal Ontology in Information Systems, Amsterdam 1998, S. 163-178.

Gangemi, Steve et al. (1996)
Gangemi, A.; Steve, G.; Giacomelli, F.: ONIONS: An Ontological Methodology for Taxonomic Know-ledge Integration. In: van der Vet, P. (Hrsg.): Proceedings of the Workshop on Ontological Engineering (in conjunction with ECAI´96), Budapest 1996, S. 1-11.
URL: http://wwwkbs.cs.utwente.nl/EcaiWorkshop/fullpapers.html, Zugriff am 18.02.2003.

Garvin (1984)
Garvin, D. A.: What Does "Product Quality" Really Mean? In: Sloan Management Review, Jg. 25 (1984), Nr. 3, S. 25-43.

Garvin (1987)
Garvin, D. A.: Competing on the Eight Dimensions of Quality. In: Harvard Business Review, Jg. 65 (1987), Nr. 6, S. 101-109.

Garvin (1988)
Garvin, D. A.: Die acht Dimensionen der Produktqualität. In: Harvard Manager, Jg. 10 (1988), Nr. 3, S. 66-74.

Genesereth, Fikes (1992)
Genesereth, M. R.; Fikes, R. E.: Knowledge Interchange Format – Version 3.0, Reference Manual. Technical Report, Computer Science Department, Stanford 1992.
URL: http://www-ksl.stanford.edu/knowledge-sharing/papers/kif.ps, Zugriff am 18.03. 2007.

Genesereth, Nilsson (1987)
Genesereth, M. R.; Nilsson, N. J.: Logical Foundations of Artificial Intelligence, Palo Alto1987.

Gerst, Hackl et al. (2001)
Gerst, M.; Hackl, H.; Liestmann, V.; Zimmermann, O.: Wege zum Wissen – Branchenweite Studie zum Wissensmanagement im Produktlebenszyklus zeigt Defizite und Handlungsfelder auf. In: QZ – Qualität und Zuverlässigkeit, Jg. 46 (2001), Nr. 1, S. 51-56.

Ghezzi, Jazayeri et al. (1991)
Ghezzi, C.; Jazayeri, M.; Mandrioli, D.: Fundamentals of Software Engineering, New Jersey 1991.

Gilbert, Strebel (1987)
Gilbert, X.; Strebel, P.: Strategies to Outpace the Competition. In: Journal of Business Strategy, Jg. 8 (1987), Nr. 1, 28-36.

Gimpel, Stolten et al. (2002)
Gimpel, B.; Stolten, D.; Dohle, H.; Mergel, J.: Fehlerfahndung – Risikomanagement mit kreativem FMEA-Verfahren. In: QZ – Qualität und Zuverlässigkeit, Jg. 47 (2002), Nr. 6, S. 646-649.

Gogoll (1994)
Gogoll, A.: Die sieben Management-Werkzeuge. In: QZ – Qualität und Zuverlässigkeit, Jg. 39 (1994), Nr. 5, S. 516-521.

Gogoll (2000)
Gogoll, A.: Service-QFD: Quality Function Deployment im Dienstleistungsbereich. In: Bruhn, M.; Stauss, B. (Hrsg.): Dienstleistungsqualität, 3. Aufl., Wiesbaden 2000, S. 363-378.

Gómez-Pérez (1994)
Gómez-Pérez, A.: From Knowledge Based Systems to Knowledge Sharing Technology: Evaluation and Assessment. Technical Report KSL 94-73, Knowledge Systems Laboratory, Stanford University, Stanford 1994.

Gómez-Pérez (2001)
Gómez-Pérez, A.: Methodologies, Tools and Languages. Where is the meeting point? Präsentation anlässlich: 5th International Protégé Workshop, Newcastle 2001, S. 1-38.
URL: http://www.schin.ncl.ac.uk/protege2001/presentations/GOMEZ-~1.PDF, Zugriff am 8.02.2003.

Gómez-Pérez, Benjamins (1999)
Gómez-Pérez, A.; Benjamins, V. R.: Overview of Knowledge Sharing and Reuse Components: Ontologies and Problem-Solving Methods. In: Benjamins, V. R.; Chandrasekaran, B.; Gómez-Pérez, A.; Guarino, N.; Uschold, M. (Hrsg.): Proceedings of the IJCAI-99 Workshop on Ontologies and Problem-Solving Methods (KRR5), Stockholm 1999, S. 1.1-1.15.
URL: http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-18/, Zugriff am 18.03.2007.

Gómez-Pérez, Fernández et al. (1996)
Gómez-Pérez, A.; Fernández, M.; de Vincente, A. J.: Towards a Method to Conceptualize Domain Ontologies. In: Proceedings of the Workshop Ontological Engineering ECAI-96, Budapest 1996, S. 41-52.

Gómez-Pérez, Fernández-López et al. (2002)
Gómez-Pérez, A.; Fernández-López, M.; Corcho, O.: Ontoweb – Technical Roadmap. Ontoweb Deliverable 1.1.2, Projektbericht Universidad Politécnica de Madrid, Madrid 2002.
URL: http://ontoweb.aifb.uni-karlsruhe.de/About/Deliverables/D1.1.2_v1_0.zip, Zugriff am 18.03.2007.

Gómez-Pérez, Fernández-López et al. (2004)
Gómez-Pérez, A.; Fernández-López, M.; Corcho, O.: Ontological Engineering: with examples from the areas of Knowledge Management, e-Commerce and the Semantic Web, London 2004.

Görz, Wachsmuth (2003)
Görz, G.; Wachsmuth, I.: Einleitung. In: Görz, G.; Rollinger, C.-R.; Schneeberger, J. (Hrsg.): Handbuch der Künstlichen Intelligenz, 4. Aufl., München 2003, S. 1-16.

Graham (1997)
Graham, P: ANSI Common Lisp, München 1997.

Grant (2002)
Grant, R. M.: Contemporary Strategy Analysis – Concepts, Techniques, Applications, 4. Aufl., Oxford 2002.

Gruber (1993)
Gruber, R.: A Translation Approach to Portable Ontology Specifications. In: Knowledge Acquisition, Jg. 5 (1993), Nr. 2, S. 199-220.

Gruber (1995)
Gruber, T.R.: Toward Principles for the Design of Ontologies Used for Knowledge Sharing. In: International Journal of Human-Computer Studies, Jg. 43 (1995), Nr. 5/6, S. 907-928.

Grüninger, Fox (1995)
Grüninger, M.; Fox, M. S.: Methodology for the Design and Evaluation of Ontologies. In: o. Hrsg.: Workshop on Basic Ontological Issues in Knowledge Sharing (IJCAI ´95), Montreal 1995, S. 1-10.
URL: http://www.eil.utoronto.ca/EIL/public/method.ps, Zugriff am 18.03.2007.

Grüninger, Lee (2002)
Grüninger, M.; Lee, J.: Ontology – Applications and Design. In: Communications of the ACM, Jg. 45 (2002), Nr. 2, S. 39-41.

Guarino (1995)
Guarino, N.: Formal Ontology, Conceptual Analysis and Knowledge Representation. In: International Journal of Human-Computer Studies, Jg. 43 (1995), Nr. 5/6, S. 625-640.
URL: http://www.loa-cnr.it/Papers/FormOntKR.pdf, Zugriff am 18.03.2007, S. 1-21.

Guarino (1998)
Guarino, N.: Formal Ontology and Information Systems. In: Guarino, N. (Hrsg.): Formal Ontology in Information Systems, Proceedings of 1st International Conference on Formal Ontologies in Information Systems (FOIS ’98), Amsterdam 1998, S. 3-15.

Guarino, Carrara et al. (1994)
Guarino, N; Carrara, M.; Giaretta, P.: An Ontology of Meta-Level Categories. In: Doyle, J.; Sandewall, E.; Torasso, P.: KR´94: Principles of Knowledge Representation and Reasoning, San Francisco 1994, S. 270-280.

Guarino, Giaretta (1995)
Guarino, N.; Giaretta, P.: Ontologies and Knowledge Bases – Towards a Terminological Clarification. Paper, Padova 1995. Geringfügig überarbeitete Version des gleichnamigen Beitrags in: Mars, N. J. I. (Hrsg.): Towards Very Large Knowledge Bases: Knowledge Building and Knowledge Sharing, Amsterdam 1995, S. 25-32. URL: http://www.loa-cnr.it/Papers/KBKS95.pdf, Zugriff am 18.03.2007, S. 1-7.

Guarino, Welty (2002)
Guarino, N.; Welty, C.: Evaluating Ontological Decisions with OntoClean. In: Communications of the ACM, Jg. 45 (2002), Nr. 2, S. 61-65.

Guarino, Welty (2002)
Guarino, N.; Welty, C.: An Overview of OntoClean. In: Staab, S.; Studer, R. (Hrsg.): Handbook on Ontologies, Berlin 2004, S. 151-171.

Guizzardi, Wagner et al. (2004)
Guizzardi, G.; Wagner, G.; Herre, H.: On the Foundations of UML as an Ontology Representation Language. In: Motta, E.; Shadbolt, N.; Stutt, A.; Gibbins, N. (Hrsg.): Engineering Knowledge in the Age of the Semantic Web, Proceedings of the 14th International Conference on Knowledge Engineering and Know-ledge Management 2004, Berlin 2004, S. 47-62.

Hartmann (1964)
Hartmann, N.: Neue Wege der Ontologie, 4. Aufl., Darmstadt 1964.

Hartmann, Janich (1996)
Hartmann, D.; Janich, P.: Methodischer Kulturalismus. Zwischen Naturalismus und Postmoderne, Frankfurt/Main 1996.

Haun (2000)
Haun, M.: Wissensbasierte Systeme, Renningen 2000.

Hennig (2001)
Hennig, B.: Prozessorientiertes Qualitätsmanagement von Dienstleistungen: ein informationswirtschaftlicher Ansatz, Dissertation Universität Karlsruhe, Wiesbaden 2001.

Herb, Herb (1998)
Herb, R.; Herb, T.; Kratzer, M.: Potentielle Fehler vollständig erfassen – Qualitative Fehlerbäume als Methode zur Kreativitätsförderung und Strukturierung von FMEA. In: QZ – Qualität und Zuverlässigkeit, Jg. 43 (1998), Nr. 2, S. 183-187.

Hesse (2002)
Hesse, W.: Ontologien. In: Informatik Spektrum, Jg. 25 (2002), Nr. 6, S. 477-480.

Holsapple, Joshi (1999)
Holsapple, C. W.; Joshi, K. D.: Knowledge selection: Concepts, issues, and technology. In: Liebowitz, J. (Hrsg.): Knowledge Management Handbook, Boca Raton 1999, S. 7-22.

Holsapple, Joshi (2002)
Holsapple, C. W.; Joshi, K. D.: A Collaborative Approach to Ontology Design. In: Communications of the ACM, Jg. 45 (2002), Nr. 2, S. 42-47.

Holsapple, Singh (2003)
Holsapple, C. W.; Singh, M.: The Knowledge Chain Model: Activities for Competitiveness.
In: Holsapple, C. W. (Hrsg.): Handbook on Knowledge Management, Band 2, Berlin 2003, S. 215-251.

Homp, Danner (2002)
Homp, C.; Danner, M.: Kernkompetenzorientiertes Management und BSC. In: Bellmann, K. (Hrsg.): Aktionsfelder des Kompetenz-Managements – Ergebnisse des II. Symposiums Strategisches Kompetenz-Mana-gement, Wiesbaden 2002, S. 439-455.

Horrocks, Patel-Schneider et al. (2003)
Horrocks, I.; Patel-Schneider, P. F.; van Harmelen, F.: From SHIQ and RDF to OWL: the making of a Web Ontology Language. In: Journal of Web Semantics: Science, Services and Agents on the World Wide Web, Jg. 1 (2003), Nr. 1, S. 7-26.

Horrocks, Patel-Schneider et al. (2004)
Horrocks, I.; Patel-Schneider, P. F.; Boley, H.; Tabet, S.; Grosof, B.; Dean, M.: SWRL: A Semantic Web Rule Language – Combining OWL and RuleML, o. O. 2004, o. S.
URL: http://www.w3.org/Submission/SWRL/, Zugriff am 18.03.2007.

Huang, Lee et al. (1999)
Huang, K.-T.; Lee, Y. W.; Wang, R. Y.: Quality Information and Knowledge, Upper Saddle River 1999.

Huang, Shi et al. (2000)
Huang, G. Q.; Shi J.; Mak, K. L.: Failure Mode and Effect Analysis (FMEA) Over the WWW. In: International Journal of Advanced Manufacturing Technology, Jg. 16 (2000), Nr. 8, S. 603-608.

Hughes, Chou et al. (1999)
Hughes, N.; Chou, E.; Price, C.; Lee, M.: Automating Mechanical FMEA Using Functional Models. In: Kumar, A.; Russell, I.: Proceedings of the 12th International Florida AI Research Society Conference Orlando, Menlo Park 1999, S. 394-398.

Hulebak, Schlosser (2002)
Hulebak, K. L.; Schlosser, W.: Hazard Analysis and Critical Control Point (HACCP) – History and Conceptual Overview. In: Risk Analysis: An International Journal, Jg. 22 (2002), Nr. 3, S. 547-552.

Husserl (1993)
Husserl, E.: Ideen zu einer reinen Phänomenologie und phänomenologischen Philosophie, 5. Aufl. Tübingen 1993.

IEEE 1074:1995
IEEE Standards Association: IEEE Standard for Developing Software Life Cycle Processes – IEEE 1074-1995. The Institute of Electrical and Electronics Engineers (Hrsg.), New York 1996.

ISO/IEC 13250:2002
Biezunski, M.; Bryan, M.; Newcomb, S. R. (Hrsg.): ISO/IEC 13250:2002 – Topic Maps: Information Technology, Document Description and Processing Languages, o.O. 2002.
URL: http://www.y12.doe.gov/sgml/sc34/document/0322_files/iso13250-2nd-ed-v2.pdf, Zugriff am 18.03.07.

Jacob (1983)
Jacob, H.: Das PIMS-Programm. In: WISU – Das Wirtschaftsstudium, Jg. 12 (1983), Nr. 2, S. 262-266.

Jacquette (2002)
Jacquette, D.: Ontology, Chesham 2002.

Jarke, Jeusfeld et al. (1996)
Jarke, M.; Jeusfeld, M.; Peters, P.; Szczurko, P.: Informationsmanagement im Qualitätskreis – ein vernetzter Ansatz. In: Pfeifer, T. (Hrsg.): Wissensbasierte Systeme in der Qualitätssicherung: Methoden zur Nutzung verteilten Wissens, Berlin 1996, S. 146-163.

Johannsen (2000)
Johannsen, C. G.: Total Quality Management in a Knowledge Management Perspective. In: The Journal of Documentation, Jg. 56 (2000), Nr. 1, S. 42-54.

Johne, Zieglowski (2000)
Johne, S.; Zieglowski, M.: Sicherheit per Matrix – Rechnergestützte Systementwicklung mit Matrix-FMEA optimiert Produktentwicklung. In: QZ – Qualität und Zuverlässigkeit, Jg. 45 (2000), Nr. 7, S. 876-882.

Johnson-Laird (1983)
Johnson-Laird, P. N.: Mental Models – Towards a Cognitive Science of Language, Inference, and Consciousness, Cambridge 1983.

Jones, Bench-Capon et al. (1998)
Jones, D.; Bench-Capon, T.; Visser, P.: Methodologies for Ontology Development, In: o Hrsg.: Proceedings IT&KNOWS Conference, XV. IFIP World Computer Congress, Budapest 1998, S. 62-75.

Jost (2005)
Jost, P.-J.: Strategisches Management. In: Bitz, M.; Domsch, M.; Ewert, R.; Wagner, F. W. (Hrsg.): Vahlens Kompendium der Betriebswirtschaftslehre, Band 2, 5. Aufl., München 2005, S. 185-246.

Kahlbrandt (2001)
Kahlbrandt, B.: Software-Engineering mit der Unified Modeling Language, 2. Aufl., Berlin 2001.

Kalbfleisch, Schellenberg et al. (2001)
Kalbfleisch, D.; Schellenberg, A.; Gröf, S.: Wissen braucht Engagement. In: QZ – Qualität und Zuverlässigkeit, Jg. 46 (2001), Nr. 2, S. 132-133.

Kalfoglou, Robertson et al. (1999)
Kalfoglou, Y.; Robertson, D.; Tate, A.: Using Meta-Knowledge at the Application Level, Technical Report, Department of Artificial Intelligence, University of Edinburgh, Edinburgh 1999.

Kalfoglou, Schorlemmer (2003)
Kalfoglou, Y.; Schorlemmer, M.: Ontology mapping: the state of the art. In: The Knowledge Engineering Review, Jg. 18 (2003), Nr. 1, S. 1-31.

Kamiske, Brauer (2003)
Kamiske, G. F.; Brauer, J.-P.: Qualitätsmanagement von A-Z, 4. Aufl., München 2003.

Kamiske, Malorny (1994)
Kamiske, G. F.; Malorny, C.: TQM – ein bestehendes Führungsmodell mit hohen Anforderungen und großen Chancen. In: Kamiske, G. F. (Hrsg.): Die hohe Schule des Total Quality Management, Berlin 1994, S. 1-18.

Kato, Shirakawa et al. (2002)
Kato, Y.; Shirakawa, T.; Hori, K.: Utilizing Fault Cases for Supporting Fault Diagnosis Task. In: Damiani, E.; Jain, L. C.; Howlett; R. J.; Ichalkaranje, N. (Hrsg.): Knowledge-based Intelligent Information Engineering Systems and Allied Technologies, Proceedings of the Sixth International Conference on Knowledge-Based Intelligent Information and Engineering Systems, Amsterdam 2002, S. 1-5.
URL: http://www.ai.rcast.u-tokyo.ac.jp/~yoshi/papers/kes2002.pdf, Zugriff am 18.03.2007.

KBSI (1994)
o. V.: KBSI (Knowledge Based Systems Inc.): The IDEF5 Ontology Description Capture Method –Overview, Texas 1994.
URL: http://www.idef.com, Zugriff am 18.03.2007.

Keller, Kuhn (2004)
Keller, C.; Kuhn, S.: Qualitäts- und Wissensmanagement – Erfolg im Doppelpack. In: MQ – Management und Qualität, Jg. 34 (2004), Nr. 9, S. 12-14.

Kersten (1994)
Kersten, G.: Fehlermöglichkeits- und -einflussanalyse (FMEA). In: Masing, W. (Hrsg.): Handbuch Qualitätsmanagement, 3. Aufl., München 1994, S. 469-490.

Kifer, Lausen et al. (1995)
Kifer, M.; Lausen, G.; Wu, J.: Logical Foundations of Object-Oriented and Frame-Based Languages. In: Journal of the Association for Computing Machinery, Jg. 42 (1995), Nr. 4, S. 741-843.

Kim, Fox (1999)
Kim, H. M.; Fox, M. S.; Grüninger, M.: An ontology for quality management – enabling quality problem identification and tracing. In: BT Technology Journal, Jg. 17 (1999), Nr. 4, S. 131-140.

Kitamura, Mizoguchi (2004)
Kitamura, Y.; Mizoguchi, R.: Ontology-Based Functional-Knowledge Modeling Methodology and Its Deployment. In: Motta, E.; Shadbolt, N.; Stutt, A.; Gibbins, N. (Hrsg.): Engineering Knowledge in the Age of the Semantic Web, Proceedings of the 14th International Conference on Knowledge Engineering and Knowledge Management 2004, Berlin 2004, S. 99-115.

Kitamura, Sano et al. (2002)
Kitamura, Y.; Sano, T.; Namba, K.; Mizoguchi, R.: A functional concept ontology and its application to automatic identification of functional structures. In: Advanced Engineering Informatics, Jg. 16 (2002), Nr. 2, S. 145-163.

Knaese, Probst (2001)
Knaese, B.; Probst, G.: Wissensorientiertes Management der Mitarbeiterfluktuation. In: Zeitschrift Führung + Organisation (zfo), Jg. 70 (2001), Nr. 1, S. 35-41.

Kneuper (2000)
Kneuper, R.: Unterstützung von Software-Prozessen durch Wissensmanagement. In: Andelfinger, U.; Herzwurm, G.; Mellis, W.; Müller-Luschnat, G. (Hrsg.): Vorgehensmodelle: Wirtschaftlichkeit, Werkzeugunterstützung und Wissensmanagement, 7. Workshop der Fachgruppe 5.11 der Gesellschaft für Informatik e.V., Aachen 2000, S. 107-124.

Knight, Luk (1994)
Knight, K.; Luk, S. K.: Building a Large Knowledge Base for Machine Translation. In: Hayes-Roth, B.; Korf, R. (Hrsg.): Proceedings of the 12th National Conference on Artificial Intelligence, Seattle 1994, S. 773-778.

Kocher (1989)
Kocher, H.: Marktgerechte Qualität: eine Betrachtung für Anbieter und Abnehmer, Bern 1989.

Koontz, O´Donnell (1972)
Koontz, H.; O´Donnell, C.: Principles of Management: An Analysis of Managerial Functions, 5. Aufl., New York 1972.

Kopka (2000)
Kopka, C.: Ein Vorgehensmodell für die Entwicklung multimedialer Lernsysteme. In: Andelfinger, U.; Herzwurm, G.; Mellis, W.; Müller-Luschnat, G. (Hrsg.): Vorgehensmodelle: Wirtschaftlichkeit, Werkzeugunterstützung und Wissensmanagement, 7. Workshop der Fachgruppe 5.11 der Gesellschaft für Informatik e.V., Aachen 2000, S. 91-105.

Krallmann (1992)
Krallmann, H.: Ein wissensbasiertes System zur fertigungsbegleitenden Qualitätssicherung für die Raumfahrt. In: Scheer, A.-W. (Hrsg.): Simultane Produktentwicklung, Forschungsbericht 4, München 1992, S. 211-232.

Kraus (2003)
Kraus, U.: ERP-OnTo-PDM: Konzept und prototypische Realisierung einer ontologiebasierten ERP/PDM Kopplung mittels XML-Technologie, Dissertation Universität Duisburg-Essen (Campus Essen), Essen 2003.

Kreikebaum (1997)
Kreikebaum, H.: Strategische Unternehmensplanung. 6. Aufl., Stuttgart 1997.

Labrou (2002)
Labrou, Y.: Agents and ontologies for e-business. In: The Knowledge Engineering Review, Jg. 17 (2002), Nr. 1, S. 81-85

Labrou, Finin (1999)
Labrou, Y.; Finin, T.: Yahoo! as an Ontology – Using Yahoo! Categories to Describe Documents. In: Gauch, S. (Hrsg.): Proceedings of the 8th International Conference on Information Knowledge Management: CIKM '99, New York 1999, S. 180-187.

Landis, Baumann (2003)
Landis, M.; Baumann, P.: Vorsprung durch Wissen – Einsatz eines konzernweiten Wissensportals in der Qualitätssicherung. In: QZ – Qualität und Zuverlässigkeit, Jg. 48 (2003), Nr. 10, S. 992-995.

Lang (1997)
Lang, K: Gestaltung von Geschäftsprozessen mit Referenzprozessbausteinen, Wiesbaden 1997.

Lau, Sure (2002)
Lau, T.; Sure, Y.: Introducing Ontology-based Skills Management at a large Insurance Company. In: Glinz, M., Müller-Luschnat, G. (Hrsg.): Modellierung 2002, Modellierung in der Praxis – Modellierung für die Praxis, Bonn 2002, S. 123-134.

Lee (2001a)
Lee, B. H.: Using FMEA models and ontologies to build diagnostic models. In: Artificial Intelligence for Engineering Design, Analysis and Manufacturing, Jg. 15 (2001), Nr. 4, S. 281-293.

Lee (2001b)
Lee, B. H.: Using Bayes belief networks in industrial FMEA modelling and analysis. In: o. V.: Proceedings of the 2001 Annual Reliability and Maintainability Symposium RAMS 2001, Philadelphia 2001, S. 7-15.

Lenat (1995)
Lenat, D.: Cyc – A large-scale Investment in Knowledge Infrastructure. In: Communications of the ACM, Jg. 38 (1995), Nr. 11, S. 33-38.

Lenat, Guha et al. (1990)
Lenat, D.; Guha, R.; Pittman, K.; Pratt, D.; Shepherd, M.: Cyc – Toward Programs with Common Sense. In: Communications of the ACM, Jg. 33 (1990), Nr. 8, S. 30-49.

Leutsch (2002)
Leutsch, M.: Unterstützung des Konstruktionsprozesses durch Integration von prozeduralem Wissen, Dissertation Universität Karlsruhe, Aachen 2002.

Linstone, Turoff (1975)
Linstone, H.; Turoff, M.: The Delphi Method: Techniques and Applications, Reading 1975.

Lorenzen (1987)
Lorenzen, P.: Lehrbuch der konstruktiven Wissenschaftstheorie, Mannheim 1987.

Luchs, Neubauer (1986)
Luchs, R. H.; Neubauer, F.-F.: Qualitätsmanagement – Wettbewerbsvorsprung durch Differenzierung. Aschaffenburg 1986.

Ludäscher, Himmeröder et al. (1998)
Ludäscher, B.; Himmeröder, R.; Lausen, G.; May, W.; Schlepphorst, C.: Managing Semistructured Data With FLORID: A Deductive Object-Oriented Perspective. In: Information Systems, Jg. 23 (1998), Nr. 8, S. 1-25.

Ludwig (1998)
Ludwig, M.: Globalisierung der Märkte: Motor oder Bremse für den Wohlstand hochentwickelter Volkswirtschaften?, Dissertation Universität Würzburg, Frankfurt a. M. 1998.

Mäckel (2001)
Mäckel, O.: Mit Blick auf´s Risiko – Software-FMEA im Entwicklungsprozess softwareintensiver technischer Systeme. In: QZ – Qualität und Zuverlässigkeit, Jg. 46 (2001), Nr. 1, S. 65-68.

Maedche (2002)
Maedche, A.: Skillmanagement der nächsten Generation. Vortrag Arbeitskreis Wissensmanagement Karlsruhe am 1.6.2002, Vortragsunterlagen, Karlsruhe 2002.

Maedche, Staab (2001)
Maedche, D.; Staab, S.: Ontology Learning for the Semantic Web. In: IEEE Intelligent Systems, Jg. 16 (2001), Nr. 2, S. 72-79.

Maedche, Staab et al. (2003)
Maedche, A.; Staab, S.; Stojanovic, N.; Studer, R.; Sure, Y.: Semantic portAL: The SEAL Approach. In: Fensel, D.; Hendler, J.; Lieberman, H.; Wahlster, W. (Hrsg.): Spinning the Semantic Web – Bringing the World Wide Web to Its Full Potential, Cambridge 2003, S. 317-359.

Maier (2002)
Maier, R.: Knowledge Management Systems: Information and Communication Technologies for Knowledge Management, Habilitationsschrift Universität Regensburg, Berlin 2004.

Marciniak, Reifer (1997)
Marciniak, J. J.; Reifer, D. J.: Software Acquisition Management. In: Reifer, D. J. (Hrsg.): Software Management, 5. Aufl., Washington 1997, S. 585-621.

May (2000)
May, W.: How to Write F-Logic Programs in Florid – A Tutorial for the Database Language F-Logic, Freiburg 2000.

McGuinness, van Harmelen (2004)
McGuinness, D. L.; van Harmelen, F. (Hrsg.): OWL, Web Ontology Language – Overview, W3C Recommendation, o. O. 2004, o. S.
URL: http://www.w3.org/TR/2004/REC-owl-features-20040210/, Zugriff am 18.03.2007.

Metzger (2000)
Metzger, J.: Matrix statt Liste – Implementierung und Praxiseinsatz der Matrix-FMEA-Methode. In: QZ – Qualität und Zuverlässigkeit, Jg. 45 (2000), Nr. 5, S. 592-596.

Miller, Beckwith et al. (1990)
Miller, G.; Beckwith, R.; Fellbaum, C.; Gross, D.; Miller, K.: Introduction to WordNet: An Online Lexical Database. In: International Journal of Lexicography, Jg. 3 (1990), Nr. 4, S. 235-244.

MIL-STD 1629A:1980
MIL-STD-1629: Procedures for performing FMECA, Revision A, Washington 1980.

Minsky (1975)
minsky, A.: Framework for Representing Knowledge. In: Winston, P. (Hrsg.): The Psychology of Computer Vision, New York 1975, S. 211-277.
URL: http://web.media.mit.edu/~minsky/papers/Frames/frames.html, Zugriff am 18.03.2007.

Mizoguchi, Ikeda (1996)
Mizoguchi, R.; Ikeda, M.: Towards Ontology Engineering. Technical Report 96-1, Institute of Scientific and Industrial Research, Osaka University, Osaka 1996.

Mizoguchi, Kozaki et al. (2000)
Mizoguchi, R.; Kozaki, K.; Sano, T.; Kitamura, Y.: Construction and Deployment of a Plant Ontology. In: Dieng, R.; Corby, O. (Hrsg.): Proceedings of the 12th European Workshop Knowledge Acquisition, Modeling and Management, Lecture Notes In Computer Science, Vol. 1937, London 2000, S. 113-128.

Mohr (1999)
Mohr, H.: Wissen – Prinzip und Ressource, Berlin 1999.

Möller, Haarslev (2003)
Möller, R.; Haarslev, V.: Description Logics for the Semantic Web: Racer as a Basis for Building Agent Systems. In: KI – Künstliche Intelligenz, Jg. 17 (2003), Nr. 3, S. 10-15.

Moody, Shanks (1994)
Moody, D.; Shanks, G.: What Makes a Good Data Model? Evaluating the Quality of Entity Relationship Models. In: Loucopoulos, P. (Hrsg.): Entity-Relationship Approach – ER ’94, Proceedings of the 13th International Conference on the Entity-Relationship Approach (Manchester), Heidelberg 1994, S. 94-111.

Moran, Carrol (1996)
Moran, T. P.; Carrol, J. M.: Overview of Design Rationale. In: Moran, T. P.; Carrol, J. M (Hrsg.): Design Rationale: Concepts, Techniques, and Use, Mahwah 1996, S. 1-20.

Moravec (1990)
Moravec, H.: Mind children: der Wettlauf zwischen menschlicher und künstlicher Intelligenz. Hamburg 1990.

Motta (1999)
Motta, E.: Reusable Components for Knowledge Modelling – Case Studies in Parametric Design Problem Solving, Amsterdam 1999.

Motta, Shum et al. (2000)
Motta, E.; Shum, S. B.; Domingue, J.: Ontology-driven document enrichment: principles, tools and applications. In: International Journal of Human-Computer Studies, Jg. 52 (2000), Nr. 5, S. 1071-1109.

Motta, Zdrahal (1998)
Motta, E.; Zdrahal, Z.: A library of problem-solving components based on the integration of the search paradigm with task and method ontologies. In: International Journal of Human-Computer Studies, Jg. 49 (1998), Nr. 4, S. 437-470.

Moynihan, Bowers et al. (2002)
Moynihan, G. P.; Bowers, J. H.; Fonseca, D. J.; Ray, P. S.: A knowledge-based approach to maintenance project planning. In: Expert Systems, Jg. 19 (2002), Nr. 2, S. 88-98.

Müller (1991)
Müller, W.: Strategisches Marketing: Ein übersehenes Wettbewerbsinstrument in der Automobilindustrie? In: DBW – Die Betriebswirtschaft, Jg. 51 (1991), Nr. 6, S. 781-799.

Myerson (1996)
Myerson, M.: Risk management processes for software engineering models, Norwood 1996.

Mylopoulos (1998)
Mylopoulos, J.: Information Modeling in the Time of the Revolution. In: Information Systems, Jg. 23 (1998), Nr. 3/4, S. 127-155.

NACMCF (1997)
National Advisory Committee on Microbiological Criteria for Foods: Hazard Analysis and Critical Control Point – Principles and Application Guidelines. In: Journal of Food Protection, Jg. 61 (1998), Nr. 9, S. 1246-1259.

Neches, Fikes et al. (1991)
Neches, R.; Fikes, R.; Finin, T.; Gruber, T.; Patil, R.; Senator, T.; Swartout, W. R.: Enabling Technology for Knowledge Sharing. In: AI Magazine, 12. Jg. (1991), Nr. 3, S. 36-56.

Nedeß, Nickel (1992)
Nedeß, C.; Nickel, J.: Wissensbasierte FMEA-Erstellung zur Unterstützung der Simultanen Produktentwicklung. In: Scheer, A.-W. (Hrsg.): Simultane Produktentwicklung, Forschungsbericht 4, München 1992, S. 277-333.

Nedeß, Nickel (1993)
Nedeß, C.; Nickel, J.: FMEA wissensbasiert erstellen. In: QZ – Qualität und Zuverlässigkeit, 38. Jg. (1993), Nr. 12, S. 689-693.

Newell (1981)
Newell, A.: The Knowledge Level: Presidential Address. In: AI Magazine, Jg. 2 (1981), Nr. 2, S. 1-20 und 33.

Ngai, Chow (1999)
Ngai, E. W. T.; Chow, D. Y. H.: ICADS: Intelligent Car Audio Design System for product planning. In: Expert Systems, Jg. 16 (1999), Nr. 1, S. 19-32.

Nickel (1992)
Nickel, J.: Technische und methodische Hilfsmittel zur Verbesserung der Fehler-Möglichkeits- und Einfluss-Analyse (FMEA). Dissertation Universität Hamburg-Harburg, Aachen 1992.

Noack (2000)
Noack, J.: Entwicklung, Einsatz und Pflege von Vorgehensmodellen in der Sparkassenorganisation. In: Andelfinger, U.; Herzwurm, G.; Mellis, W.; Müller-Luschnat, G. (Hrsg.): Vorgehensmodelle: Wirtschaftlichkeit, Werkzeugunterstützung und Wissensmanagement, 7. Workshop der Fachgruppe 5.11 der Gesellschaft für Informatik e.V., Aachen 2000, S. 1-16.

Nonaka, Takeuchi (1995)
Nonaka, i.; Takeuchi, H.: The Knowledge-Creating Company – How Japanese Companies Create the Dynamics of Innovation, New York 1995.

Nonaka, Takeuchi (1997)
Nonaka, I.; Takeuchi, H.: Die Organisation des Wissens – Wie japanische Unternehmen eine brachliegende Ressource nutzbar machen, Frankfurt/Main 1997.

North (2002)
North, K.: Wissensorientierte Unternehmensführung – Wertschöpfung durch Wissen, 3. Aufl., Wiesbaden 2002.

Norton (2003)
Norton, C.: HACCP – Developing and verifying a flow diagram for food production. In: Food Management, Jg. 38 (2003), Nr. 5, S. 80-81.

Noy (2004)
Noy, N. F.:Tools for Mapping and Merging Ontologies. . In: Staab, S.; Studer, R. (Hrsg.): Handbook on Ontologies, Berlin 2004, S. 365-384.

Noy, Fergerson et al. (2000)
Noy, N. F.; Fergerson, R. W.; Musen, M. A.: The Knowledge Model of Protégé-2000: Combining Interoperability and Flexibility. In: Dieng, R.; Corby, O. (Hrsg.): Knowledge Engineering and Knowledge Management. Methods, Models, and Tools: 12th International Conference (EKAW ´00), Lecture Notes in Artificial Intelligence, Nr. 1937, Berlin 2000, S. 17-32.

Obrst, Liu (2003)
Obrst, L.; Liu, H.: Knowledge Representation, Ontological Engineering, and Topic Maps. In: Park, J. (Hrsg.); Hunting, S. (technischer Hrsg.): XML Topic Maps – Creating and Using Topic Maps for the Web, Boston 2003, S. 103-148.

Oesterreich (2002)
Oesterreich, B.: Die UML-Kurzreferenz für die Praxis, 2. Aufl., München 2002.

OMG (2003)
Object Management Group (OMG): Unified Modeling Language (UML) – Version 1.5, o.O. 2003.
URL: http://www.omg.org/technology/documents/formal/uml.htm, Zugriff am 18.03.2007.

Ontoprise (2004)
Ontoprise GmbH (Hrsg.): How to Write F-Logic Programs – A Tutoriarl for the Language F-Logic, Karlsruhe 2004. URL: http://www.ontoprise.de/documents/tutorial_flogic.pdf, Zugriff am 18.03.2007.

Opwis (1992)
Opwis, K.: Kognitive Modellierung: zur Verwendung wissensbasierter Systeme in der psychologischen Theoriebildung, Habilitationsschrift Universität Freiburg 1990, Bern 1992.

Ostermayer (2001)
Ostermayer, R.: Pragmatisch-situative Wissensrepräsentation – ein Baustein für das Wissensmanagement, Dissertation Universität Karlsruhe, Aachen 2001.

Owsnicki-Klewe, von Luck et al. (2003)
Owsnicki-Klewe, B.; von Luck, K.; Nebel, B.: Wissensrepräsentation und Logik – Eine Einführung. In: Görz, G.; Rollinger, C.-R.; Schneeberger, J.(Hrsg.): Handbuch der Künstlichen Intelligenz, 4. Aufl., München 2003, S. 153-197.

Pagel, Six (1991)
Pagel, B.-U.; Six, H.-W.: Software Engineering – Die Phasen der Softwareentwicklung, Bonn 1994.

Park, Hunting (2003)
Park, J. (Hrsg.); Hunting, S. (technischer Hrsg.): XML Topic Maps – Creating and Using Topic Maps for the Web, Boston 2003.

Parsons, Wand (2000)
Parsons, J.; Wand, Y.: Emancipating Instances from the Tyranny of Classes in Information Modelling. In: ACM Transactions on Database Systems, Jg. 25 (2000), Nr. 2, S. 228-268.

Patel-Schneider, Hayes et al. (2004)
Patel-Schneider, P. F.; Hayes, P.; Horrocks, I.: OWL Web Ontology Language – Semantics and Abstract Syntax, o.O. 2004, o.S.
URL: http://www.w3.org/TR/owl-semantics/, Zugriff am 18.03.2007.

Paulic, Starke (1994)
Paulic, R.; Starke, G.: Rechnergestützte Fehlermöglichkeits- und Einflußanalyse (FMEA) – Methodik und Softwaremarkt, Frankfurt am Main 1994.

Pease, Niles et al. (2002)
Pease, A.; Niles, I.; Li, J.: The Suggested Upper Merged Ontology – A large Ontology for the Semantic Web and its Applications. In: AAAI (Hrsg.): Working Notes of the Workshop on Ontologies and the Semantic Web, Edmonton 2002, S. 7-10.

Pfeifer (2001)
Pfeifer, T.: Qualitätsmanagement – Strategien, Methoden, Techniken, 3. Aufl., München 2001.

Pfeifer (2002)
Pfeifer, T.: Quality Management – Strategies, Methods, Techniques, München 2002.

Pfeifer, Grob et al. (1996)
Pfeifer, T.; Grob, R.; Klonaris, P.: Wissensbasierte Fehleranalyse. In: Pfeifer, T. (Hrsg.): Wissensbasierte Systeme in der Qualitätssicherung: Methoden zur Nutzung verteilten Wissens, Berlin 1996, S. 96-112.

Pfeifer, Reinecke et al. (1998)
Pfeifer, T.; Reinecke, R.; Grebner, H.: FMEA-Wissen transparent machen. In: QZ – Qualität und Zuverlässigkeit, Jg. 43 (1998), Nr. 10, S. 1210-1213.

Pfeifer, Reinecke et al. (2000)
Pfeifer, T.; Reinecke, R.; Dahmen, J.; Kaminski, V.; Schneider, K.: Zuverlässig kuppeln – Unternehmensübergreifende Prozessabsicherung in der Automobilzulieferindustrie. In: QZ – Qualität und Zuverlässigkeit, Jg. 45 (2000), Nr. 8, S. 994-997.

Pfeifer, Scheermesser et al. (2000)
Pfeifer, T.; Scheermesser, S.; Lorenzi, P.: Verborgene Potentiale erschließen. In: QZ – Qualität und Zuverlässigkeit, Jg. 45 (2000), Nr. 3, S. 277-279.

Pieper (1998)
Pieper, K.-P.: Nie wieder Angst vor dem Störfall – FMEAs für die Erstellung von Notfallplänen nutzen. In: QZ – Qualität und Zuverlässigkeit, Jg. 43 (1998), Nr. 8, S. 940-944.

Pierlein (1995)
Pierlein, T.: Wiederverwendung von Commonsense Ontologien im Knowledge Engineering – Methoden und Werkzeuge, Dissertation Universität Karlsruhe, Sankt Augustin 1995.

Pinto, Martins (2004)
Pinto, H. S.; Martins J. P.: Ontologies: How can They be Built? In: Knowledge and Information Systems, Jg. 6 (2004), Nr. 4, S. 441-464.

Pocsai (2000)
Pocsai, Z.: Ontologiebasiertes Wissensmanagement in der Produktentwicklung, Dissertation Universität Karlsruhe, Aachen 2000.

Pomberger, Blaschek (1993)
Pomberger, G.; Blaschek, G.: Software-Engineering – Grundlagen des Software Engineering: Prototyping und objektorientierte Software-Entwicklung, München 1993.

Porter (1999)
Porter, M. E.: Wettbewerbsstrategie – Methoden zur Analyse von Branchen und Konkurrenten, 10. Aufl., Frankfurt/Main 1999.

Probst, Raub et al. (2003)
Probst, G.; Raub, S.; Romhardt, K.: Wissen managen: wie Unternehmen ihre wertvollste Ressource optimal nutzen, 4. Aufl., Frankfurt/Main 2003.

Proff (1997)
Proff, H.: Hybride Strategien – Unternehmensstrategien zur Sicherung des Überlebens. In: WiSt – Wirtschaftswissenschaftliches Studium, Jg. 26 (1997), Nr. 6, S. 305-307.

Puppe (1991)
Puppe, F.: Einführung in Expertensysteme, 2. Aufl., Berlin 1991.

Puppe, Stoyan et al. (2003)
Puppe, F.; Stoyan, H.; Studer, R.: Knowledge Engineering. In: Görz, G.; Rollinger, C.-R.; Schneeberger, J. (Hrsg.): Handbuch der Künstlichen Intelligenz, 4. Aufl., München 2003, S. 599-641.

Puri (2003)
Puri, W.: Semantische Funktionsmodellierung als Basis für effizientes Product Life Cycle Management. Fortschritt-Berichte VDI, Reihe 16, Band 160, Dissertation Friedrich-Alexander-Universität Erlangen-Nürnberg, Düsseldorf 2003.

Rabbitt, Bergh (1998)
Rabbitt, J. T.; Bergh, P. A.: The QS-9000 book: The fast track to compliance, New York 1998.

Ranjan, Glassey et al. (2002)
Ranjan, A; Glassey, J.; Montague, G.; Mohan, P.: From process experts to a real-time knowledge-based system. In: Expert Systems, Jg. 19 (2002), Nr. 2, S. 69-79.

Raphael (2002)
Raphael, T.: HRMS Gets Easier, Better for Smaller Companies. In: Workforce, Jg. 81 (2002), Nr. 2, S. 46-49.

Redeker, Sauer et al. (2002)
Redeker, G.; Sauer, R.; Keunecke, L.; Meyer, C.: Keine Fehler – kaum Kosten, System-FMEA-Prozess in der automobilen Produktionsplanung. In: QZ – Qualität und Zuverlässigkeit, Jg. 47 (2002), Nr. 10, S. 1029-1032.

Reiter (1997)
Reiter, C.: Toolbasierte Referenzmodellierung – State-of-the-Art und Entwicklungstrends. In: Becker, J.; Rosemann, M.; Schütte, R. (Hrsg.): Entwicklungsstand und Entwicklungsperspektiven der Referenzmodellierung, Proceedings-Band 10.3.1997, Münster 1997, S. 34-45.
URL: http://www.wi.uni-muenster.de/inst/arbber/ab52.pdf, Zugriff am 18.03.2007.

Reiter (1999)
Reiter, C.: Toolbasierte Referenzmodellierung – State-of-the-Art und Entwicklungstrends. In: Becker, J.; Rosemann, M.; Schütte, R. (Hrsg.): Referenzmodellierung – State-of-the-Art und Entwicklungsperspektiven, Heidelberg 1999, S. 45-68.

Rosemann, Schütte (1999)
Rosemann, M.; Schütte, R.: Multiperspektivische Referenzmodellierung. In: Becker, J.; Rosemann, M.; Schütte, R. (Hrsg.): Referenzmodellierung – State-of-the-Art und Entwicklungsperspektiven, Heidelberg 1999, S. 22-44.

Rothenfluh, Gennari et al. (1994)
Rothenfluh, T.; Gennari, J.; Eriksson, H.; Puerta, A.; Tu, S.; Musen, M.: Reusable Ontologies, Know-ledge-Acquisition Tools, and Performance Systems: PROTÉGÉ-II Solutions to Sisyphus-2. In: Gaines, B. (Hrsg.): Proceedings of the 8th Banff Knowledge Acquisition for Knowledge-Based Systems Workshop, Banff 1994, S. 43.1-43.30.

Royce (1970)
Royce, W. W.: Managing the Development of Large Software Systems, Concepts and Techniques. In: o. Hrsg.: Proceedings of the IEEE WESCON, Los Angeles 1970, S. 1-9.

Rumbaugh, Jacobson et al. (2005)
Rumbaugh, J.; Jacobson, I.; Boogh, G.: The Unified Modeling Language Reference Manual – Second Edition, Boston 2005.

Ruta (1999)
Ruta, A.: Fehlermöglichkeits- und Einflußanalyse (FMEA) für die Produktionslogistik. Fortschritt-Berichte VDI, Reihe 2 Fertigungstechnik Nr. 518, Dissertation Universität Hannover, Düsseldorf 1999.

Ruth (2000)
Ruth, W.: Wissen transferieren und Handeln ermöglichen – welchen Anteil Vorgehensmodelle dabei leisten können. In: Andelfinger, U.; Herzwurm, G.; Mellis, W.; Müller-Luschnat, G. (Hrsg.): Vorgehensmodelle: Wirtschaftlichkeit, Werkzeugunterstützung und Wissensmanagement, 7. Workshop der Fachgruppe 5.11 der Gesellschaft für Informatik e.V., Aachen 2000, S. 141-155.

Ryle (1949)
Ryle, G.: The Concept of Mind, New York 1949.

Ryle (1969)
Ryle, G.: Der Begriff des Geistes, Stuttgart 1969.

SAE J1379:2002
Society of Automotive Engineers (Hrsg.): Potential Failure Mode and Effects Analysis in Design (Design FMEA) and Potential Failure Mode and Effects Analysis in Manufacturing and Assembly Processes (Process FMEA) and Effects Analysis for Machinery (Machinery FMEA), SAE Handbook 2003, Warrendale 2003, S. 1-57.

Scacchi (2001)
Scacchi, W.: Process Models in Software Engineering. In: Marciniak, J. (Hrsg.): Encyclopaedia of Software Engineering, 2. Aufl., Wiley 2002, S. 993-1005.

Scheer (1992)
Scheer, A.-W.: Expertensystem zur konstruktionsbegleitenden Kalkulation. In: Scheer, A.-W. (Hrsg.): Simultane Produktentwicklung, Forschungsbericht 4, München 1992, S. 369-389.

Scheer (1997)
Scheer, A.-W.: ARIS – House of Business Engineering: Konzept zur Beschreibung und Ausführung von Referenzmodellen. In: Becker, J.; Rosemann, M.; Schütte, R. (Hrsg.): Entwicklungsstand und Entwicklungsperspektiven der Referenzmodellierung, Proceedings-Band 10.3.1997, Münster 1997, S. 3-15.
URL: http://www.wi.uni-muenster.de/inst/arbber/ab52.pdf, Zugriff am 18.03.2007.

Scheer (1999)
Scheer, A.-W.: „ARIS – House of Business Engineering" – Konzept zur Beschreibung und Ausführung von Referenzmodellen. In: Becker, J.; Rosemann, M.; Schütte, R. (Hrsg.): Referenzmodellierung – State-of-the-Art und Entwicklungsperspektiven, Heidelberg 1999, S. 1-21.

Schiegg, Viertlböck et al. (1999)
Schiegg, H.; Viertlböck, M.; Kraus, T.: Prozeßbegleitend und frühzeitig – System-Produkt-FMEA mit objektiver Kennzahlenbildung bei einem Automobilzulieferer. In: QZ – Qualität und Zuverlässigkeit, Jg. 44 (1999), Nr. 7, S. 879-884.

Schindler (2001)
Schindler, M.: Wissensmanagement in der Projektabwicklung – Grundlagen, Determinanten und Gestaltungskonzepte eines ganzheitlichen Projektwissensmanagements, 2. Aufl., Köln 2001.

Schlenoff, Ivester et al. (1998)
Schlenoff, C.; Ivester, R.; Knutilla, A.: A Robust Ontology for Manufacturing Systems Integration. In (o. Hrsg.): Proceedings of 2nd International Conference on Engineering Design and Automation, Maui 1998, S. 1-6.

Schloske (1999)
Schloske, A.: Ein Modell zur EDV-integrierten Fehlermöglichkeits- und einflußanalyse (FMEA) in der Arbeitsplanung, Prüfplanung und Fertigung, Dissertation Universität Stuttgart, Heimsheim 1999.

Schmidt, Minges et al. (1991)
Schmidt, J.; Minges, R.; Asteriadis, N.; Müller, A.: FMEA – eine Chance für den Mittelstand. In: QZ – Qualität und Zuverlässigkeit, Jg. 36 (1991), Nr. 1, S. 27-30.

Schnurr, Sure et al. (2000)
Schnurr, H.; Sure, Y.; Studer, R.; Akkermanns, H.: On-To-Knowledge Methodology – Baseline Version. On-To-Knowledge Deliverable D-15, Institut AIFB, Universität Karlsruhe, Karlsruhe 2000.
URL: http://www.ontoknowledge.org/countd/countdown.cgi?del15.pdf, Zugriff am 18.03.2007.

Schöfer (1999)
Schöfer, A.: Unterstützung kognitiver Arbeitsaufgaben in Produktionssystemen, Dissertation Universität Hannover, Hannover 1999.

Schöning (1992)
Schöning, U.: Logik für Informatiker, 3. Aufl., Mannheim 1992.

Schöning (2000)
Schöning, U.: Logik für Informatiker, 5. Aufl., Heidelberg 2000.

Schreiber, Akkermans et al. (2001)
Schreiber, G.; Akkermans, H.; Anjewierden, A.; de Hoog, R.; Shadbolt, N.; Van de Velde, W.; Wielinga, B.: Knowledge Engineering and Management: The CommonKADS Methodology, 2. Aufl., Massachusetts 2001.

Schreiber; Wielinga et al. (1994)
Schreiber, G.; Wielinga, B; de Hoog, R.; Akkermans, H.; Van de Velde, W.: CommonKADS: A Comprehensive Methodology for KBS Development. In: IEEE Expert, Jg. 9 (1994), Nr. 6, S. 28-37.

Schreiber, Wielinga et al. (1995)
Schreiber, G.; Wielinga, B.; Jansweijer, W.: The KACTUS View on the 'O' Word. In: o. Hrsg.: Workshop on Basic Ontological Issues in Knowledge Sharing (IJCAI '95), Montreal 1995, S. 1-10.
URL: http://www.swi.psy.uva.nl/usr/Schreiber/papers/Schreiber95a.pdf, Zugriff am 18.03.2007.

Schub, Stark (1985)
Schub, A.; Stark, K.: Life Cycle Cost von Bauobjekten, Köln 1985.

Schütte (1997)
Schütte, R.: Die neuen Grundsätze ordnungsmäßiger Modellierung. Paper zum Forschungsforum '97, Leipzig 1997.

Schütte (1998)
Schütte, R.: Grundsätze ordnungsmäßiger Referenzmodellierung – Konstruktion konfigurations- und anpassungsorientierter Modelle, Dissertation Universität Münster, Wiesbaden 1998.

Schütte, Zelewski (2002)
Schütte, R.; Zelewski, S.: Epistemological Problems in Working with Ontologies. In: Callaos, N.; Porter, J.; Rishe, N. (Hrsg.): The 6th World Multiconference on Systemics, Cybernetics and Informatics (SCI 2002), 14.-18.07.2002 in Orlando, USA, Proceedings, Vol. VII: Information Systems Development II, Orlando 2002, S. 161-167.

Sedbrook (2001)
Sedbrook, T.: Integrating XML business forms and rule-based agent technologies. In: Expert Systems, Jg. 18 (2001), Nr. 5, S. 250-260.

Seidlmeier (2002)
Seidlmeier, H.: Prozessmodellierung mit ARIS® – Eine beispielorientierte Einführung für Studium und Praxis, Braunschweig 2002.

Shanks (1997)
Shanks, G.G.: Conceptual Data Modelling. An Empirical Study of Expert and Novice Data Modellers. In: Australian Journal of Information Systems, Jg. 4 (1997), Nr. 2, S. 63-73.

Siefkes (1990)
Siefkes, D.: Formalisieren und Beweisen – Logik für Informatiker, Braunschweig 1990.

Silberbusch (1990)
Silberbusch, P.: Methoden der Wissensrepräsentation. In: Behrendt, R. (Hrsg.): Angewandte Wissensverarbeitung, München 1990, S. 167-181.

Smith, Becker (1997)
Smith, S.; Becker, M.A.: An Ontology for Constructing Scheduling Systems. In: Farquhar, A. (Hrsg.): Ontological engineering : papers from the 1997 AAAI Symposium, Menlo Park 1997, S. 120-129.

Sommerville (2001)
Sommerville, I.: Software Engineering, 6. Aufl., Edinburgh 2001.

Sowa (1984)
Sowa, J. F.: Conceptual Structures: Information Processing in Mind and Machine, Reading 1984.

Sowa (2000)
Sowa, J. F.: Knowledge Representation – Logical, Philosophical, and Computational Foundations, Pacific Grove 2000.

Sponner, Hoffmann et al. (2000)
Sponner, B.; Hoffmann, J.; Garbas, M.: Fehler erkannt, Kosten gebannt – Ganzheitliche System-FMEA unter Moderation senkt Kosten. In: QZ – Qualität und Zuverlässigkeit, Jg. 45 (2000), Nr. 10, S. 1279-1284.

Spur, Krause (1997)
Spur, G.; Krause, F.-L.: Das virtuelle Produkt: Management der CAD-Technik, München 1997.

Staab (2002)
Staab, S.: Wissensmanagement mit Ontologien und Metadaten. In: Informatik Spektrum, Jg. 25 (2002), Nr. 3, S. 194-209.

Staab, Schnurr (1999)
Staab, S.; Schnurr, H.-P.: Knowledge and Business Processes: Approaching an Integration. In: Dieng, R.; Matta, N. (Hrsg.): Knowledge Management and Organisational memories, Boston 2002, S. 75-88.

Staab, Studer (2004)
Staab, S.; Studer, R. (Hrsg.): Handbook on Ontologies, Berlin 2004.

Stader (1996)
Stader, J.: Results of the Enterprise Project. In: SGES (Hrsg.): Proceedings of the 16th Annual Conference of the British Computer Society Specialist Group on Expert Systems, Cambridge 1996, o. S.
URL http://www.aiai.ed.ac.uk/project/enterprise/, Zugriff am 18.03.2007.

Stamatis (2003)
Stamatis, D. H.: Failure Mode and Effect Analysis – FMEA from Theory to Execution, 2. Aufl., Milwaukee 2003.

Steinmann (2000)
Steinmann, C.: Der Faktor Mensch und Prozeßstandardisierung. In: Andelfinger, U.; Herzwurm, G.; Mellis, W.; Müller-Luschnat, G.: Vorgehensmodelle: Wirtschaftlichkeit, Werkzeugunterstützung und Wissensmanagement, Aachen 2000, S. 17-30.

Stockinger (1989)
Stockinger, K.: FMEA – Ein Erfahrungsbericht. In: QZ – Qualität und Zuverlässigkeit, Jg. 34 (1989), Nr. 3, S. 155-158.

Strang (2003)
Strang, T.: Vergleich von Wissensmodellen. Technical Report IB 554-03/02 des DLR, Oberpfaffenhofen 2003.
URL: http://www.kn.op.dlr.de/~strang/paper/reports/wissmod.pdf, Zugriff am 6.06.2004.

Strube, Habel et al. (2003)
Strube,G.; Habel, C.; Konieczny, L.; Hemforth, B.: Kognition. In: Görz, G.; Rollinger, C.-R.; Schneeberger, J. (Hrsg.): Handbuch der Künstlichen Intelligenz, 4. Aufl., München 2003, S. 19-72.

Struckmann (2002)
Struckmann, P.: Entwicklung eines wissensbasierten Informationssystems zur thermischen Umwandlung fester Brennstoffe, Dissertation Ruhr-Universität Bochum, Aachen 2002.

Strunk (2000)
Strunk, H.-M.: Lektion gelernt – Automobilzulieferer nutzt softwaregestützt FMEA-Wissen für Folgeprojekte. In: QZ – Qualität und Zuverlässigkeit, Jg. 45 (2000), Nr. 5, S. 574-578.

Struss (2004)
Struss, P.: Modellbasierte Systeme und qualitative Modellierung. In: Görz, G.; Rollinger, C.-R.; Schneeberger, J. (Hrsg.): Handbuch der Künstlichen Intelligenz, 4. Aufl., München 2003, S. 431-490.

Struss, Heller et al. (2000)
Struss, P.; Heller, u.; Malik, A.; Sachenbacher, M.: Modellbasierte Werkzeuge für Diagnose und Fehleranalyse von Fahrzeugsubsystemen. In: Hotz, L.; Struss, P.; Guckenbiehl, T. (Hrsg.): Intelligent Diagnosis in Industrial Applications, Aachen 2000, S. 17-40.

Stuckenschmidt (2003)
Stuckenschmidt, H.: Query Processing on the Semantic Web. In: KI – Künstliche Intelligenz, Jg. 17 (2003), Nr. 3, S. 22-26.

Stuckenschmidt, van Harmelen (2005)
Stuckenschmidt, H.; van Harmelen, F.: Information Sharing on the Semantic Web, Berlin 2005.

Studer, Abecker et al. (1999)
Studer, R.; Abecker, A.; Decker, S.: Informatik-Methoden für das Wissensmanagement. In: Lausen, G.; Oberweis, A.; Schlageter, G. (Hrsg.): Angewandte Informatik und Formale Beschreibungsverfahren, Stuttgart 1999, S. 263-274.

Studer, Benjamins et al. (1998)
Studer, R.; Benjamins, V. R.; Fensel, D.: Knowledge Engineering: Principles and Methods. In: IEEE Transactions on Data and Knowledge Engineering, Jg. 25 (1998), Nr. 1/2, S. 161-197.

Studer, Fensel et al. (1999)
Studer, R; Fensel, D.; Decker, S.; Benjamins, V.: Knowledge Engineering: Survey and Future Directions. In: Puppe, F. (Hrsg.): Knowledge-Based Systems: Survey and Future Directions, Proceedings of the 5th Biannual German Conference on Knowledge-Based Systems, Würzburg 1999, S. 1-23.

Sure (2002)
Sure, Y.: On-To-Knowledge – Ontology based Knowledge Management Tools and their Application. In: KI – Künstliche Intelligenz, Jg. 14 (2002), Nr. 1, S. 35-38.

Sure (2003)
Sure, Y.: Methodology, Tools and Case Studies for Ontology based Knowledge Management, Dissertation Universität Karlsruhe, Karlsruhe 2003.

Sure, Erdmann et al. (2002)
Sure, Y.; Erdmann, M.; Angele, J.; Staab, S.; Studer, R.; Wenke, D.: OntoEdit: Collaborative Ontology Engineering for the Semantic Web. In: Horrocks, I.; Hendler, J. A. (Hrsg.): First International Semantic Web Conference (ISWC02), Lecture Notes in Computer Science, Nr. 2342, Berlin 2002, S. 221-235.

Sure, Maedche et al. (2000)
Sure, Y.; Maedche, A.; Staab, S.: Leveraging Corporate Skill Knowledge – From Pro-Per to OntoProPer. In: Mahling, D.; Reimer, U. (Hrsg.): Proceedings of the Third International Conference on Practical Aspects of Knowledge Management, Basel 2000, S. 22.1-22.9.

Sure, Staab et al. (2004)
Sure, Y.; Staab, A.; Studer, R.: On-To-Knowledge Methodology (OTKM). In: Staab, S.; Studer, R. (Hrsg.): Handbook on Ontologies, Berlin 2004, S. 117-132.

Sure, Studer (2002)
Sure, Y.; Studer, R.: On-To-Knowledge Methodology – final version. On-To-Knowledge deliverable D-18, Institut für AIFB, Universität Karlsruhe, Karlsruhe 2002.
URL: http://www.ontoknowledge.org/downl/del18.pdf, Zugriff am 18.03.2007.

Swartout, Patil et al. (1996)
Swartout, B.; Patil, R.; Knight, K.; Russ, T.: Toward distributed use of large-scale ontologies. In: Gaines B.; Musen, M. (Hrsg.): Proceedings of the 10th Knowledge Acquisition for Knowledge-Based Systems Workshop (KAW '96), Alberta 1996, S. 32.1-32.19.

Swartout, Patil et al. (1997)
Swartout, B.; Patil, R.; Knight, K.; Russ, T.: Toward Distributed Use of Large-Scale Ontologies. In: Farquhar, A.; Gruninger, M. (Hrsg.): Ontological Engineering – Papers from the 1997 AAAI Symposium, Menlo Park 1997, S. 138-148.

Tammler, Eschborn (1998)
Tammler, U.; Eschborn, S.: Umweltschutz im Betrieb verbessern – durch den Einsatz von FMEA zusammen mit anderen Techniken. In: QZ – Qualität und Zuverlässigkeit, Jg. 43 (1998), Nr. 3, S. 304-307.

Thacker, Sheth et al. (2003)
Thacker, S.; Sheth, A.; Patel, S.: Complex Relationships for the Semantic Web. In: Fensel, D.; Hendler, J.; Lieberman, H.; Wahlster, W. (Hrsg.): Spinning the Semantic Web – Bringing the World Wide Web to Its Full Potential, Cambridge 2003, S. 279-315.

Theden (1997)
Theden, P.: Analyse der Rentabilität von Qualitätstechniken, Dissertation Technische Universität Berlin, Berlin 1997.

Theobald (2003)
Theobald, A.: An Ontology for Domain-oriented Semantic Similarity Search on XML Data. In: Weikum, G.; Schöning, H.; Rahm, E. (Hrsg.): 10. GI-Fachtagung Datenbanksysteme für Business, Technologie und Web (BTW), Lecture Notes in Informatics (LNI), Vol. 26, Bonn 2003, S. 217-226.

Thiele (1997)
Thiele, M.: Kernkompetenzorientierte Unternehmensstrukturen – Ansätze zur Neugestaltung von Geschäftsbereichsorganisationen, Dissertation Universität Leipzig, Wiesbaden 1997.

Timpe, Seibicke (2000)
Timpe, K.; Seibicke, V.: Fehlersuche erweitern – Lernerfolge steigern. In: QZ – Qualität und Zuverlässigkeit, Jg. 45 (2000) Ausg. 11, S. 1432-1438.

Tsou, Kao (2003)
Tsou, J.-C.; Kao, H.-P.: The Implementation of E-Commerce Ontology. In: International Journal of The Computer, The Internet and Management, Jg. 11 (2003), Nr. 1, S. 1-16.

Uitermark (2001)
Uitermark, H.: Ontology-based geographic data set integration, Dissertation Universität Enschede, Deventer 2001.

Ullman (2002)
Ullman, D. G.: Toward the ideal mechanical engineering design support system. In: Research in Engineering Design, Jg. 13 (2002), Nr. 2, S. 55-64.

Uschold (1996a)
Uschold, M.: Converting an Informal Ontology into Ontolingua: Some Experiences. In: o. Hrsg.: Proceedings of the Workshop on Ontological Engineering (in conjunction with ECAI ´96), Budapest 1996, o. S. Auch erschienen als AIAI Technical Report 192, S. 1-1.

Uschold (1996b)
Uschold, M.: Building Ontologies: Towards A Unified Methodology. In: 16th Annual Conference of the British Computer Society Specialist Group on Expert Systems, Cambridge 1996, o. S.
Auch erschienen als AIAI Technical Report 197, S. 1-18.

Uschold, Grüninger (1996)
Uschold, M.; Grüninger, M.: Ontologies: Principles, Methods and Applications. In: Knowledge Engineering Review, Jg. 11 (1996), Nr. 2, S. 93-155. Auch erschienen als AIAI Technical Report 191, S. 1-55 (in dieser Arbeit referenziert).

Uschold, King (1995)
Uschold, M.; King, M.: Towards a Methodology for Building Ontologies. In: o. Hrsg.: Workshop on Basic Ontological Issues in Knowledge Sharing (in Verbindung mit IJCAI-95), Montreal 1995, o. S.
Auch erschienen als AIAI Technical Report 183, S. 1-13.

Uschold, King et al. (1998)
Uschold, M.; King, M.; Moralee, S.; Zorgios, Y.: The Enterprise Ontology. In: The Knowledge Engineering Review, Special Issue on Putting Ontologies to Use, Jg. 13 (1998), Nr. 1, S. 31-89.

Vámos (1998)
Vámos, T.: Knowledge Representation. In: Liebowitz, J. (Hrsg.): The Handbook of Applied Expert Systems, Boca Raton 1998, S. 3.1-3.22.

van Gelder, Ross et al. (1991)
van Gelder, A.; Ross, K.; Schlipf, J. S.: The Well-Founded Semantics for General Logic Programs. In: Journal of the ACM, Jg. 38 (1991), Nr. 3, S. 620-650.

van Heijst, Schreiber et al. (1997)
van Heijst, G.; Schreiber, A. T.; Wielinga, B. J.: Using explicit ontologies in KBS development. In: International Journal Human-Computer Studies, Jg. 46 (1997), Nr. 2/3 , S. 183-292.

van Hoof, Fillies (2003)
van Hoof, A.; Fillies, C.: Das semantische Unternehmensprozessweb – Aufgaben- und rollengerechte Informationsversorgung durch vorgebaute Informationsräume. In: KI – Künstliche Intelligenz, Jg. 17 (2003), Nr. 4, S. 50-53.

van Vliet (2000)
van Vliet, H.: Software Engineering – Principles and Practice, 2. Aufl., Chichester 2000.

Vanwelkenhuysen, Mizoguchi (1995)
Vanwelkenhuysen, J.; Mizoguchi, R.: Ontologies and guidelines for modeling digital systems. In: Gaines, B. (Hrsg.): Proceedings of the 9th Knowledge Acquisition for Knowledge-Based Systems Workshop, Banff 1995, S. 4.1-4.20.

Vargas-Vera, Motta (2004)
Vargas-Vera, M.; Motta, E.: AQUA – Ontology-Based Question Answering System. In: Monroy, R.; Arroyo-Figueroa, G. (Hrsg.): MICAI 2004: Advances in Artificial Intelligence: Third Mexican International Conference on Artificial Intelligence, Lecture Notes in Computer Science, Vol. 2972, Berlin 2004, S. 468-477.

Vasconcelos, Kimble et al. (2000)
Vasconcelos, J.; Kimble, C.; Gouveia, F.; Kudenko, D.: A Group Memory System for Corporate Know-ledge Management: An Ontological Approach. In Remenyi, D. (Hrsg.): 1st European Conference on Know-ledge Management, Bled 2000, S. 91-99.

VDA-Band 4 (2003)
Verband der Automobilindustrie e.V.: Sicherung der Qualität vor Serieneinsatz – System-FMEA. VDA-Band 4: Qualitätsmanagement in der Automobilindustrie, Frankfurt 2003.

VDI 3822-1:1984
Verein Deutscher Ingenieure: Schadensanalyse – Grundlagen, Begriffe und Definitionen; Ablauf einer Schadensanalyse, VDI-Richtlinie 3822, Blatt 1, Düsseldorf 1984.

VDI/VDE 2621:1996
Verein Deutscher Ingenieure/ Verband Deutscher Elektrotechniker: Qualitätsmanagement mit Wissensbasierten Systemen – Anwenderleitfaden. VDI/VDE-Richtlinie 2621, Entwurf, Düsseldorf 1996.

Verlage (1998a)
Verlage, M.: Vorgehensmodelle und ihre Formalisierung. In: Kneuper, R.; Müller-Luschnat, G.; Oberweis, A. (Hrsg.): Vorgehensmodelle für die betriebliche Anwendungsentwicklung, Stuttgart 1998, S. 60-76.

Verlage (1998b)
Verlage, M.: Modellierungssprachen für Vorgehensmodelle. In: Kneuper, R.; Müller-Luschnat, G.; Oberweis, A. (Hrsg.): Vorgehensmodelle für die betriebliche Anwendungsentwicklung, Stuttgart 1998, S. 76-94.

V-Modell (1997)
Allgemeiner Umdruck Nr. 250/1: Vorgehensmodell – Entwicklungsstandard für IT-Systeme des Bundes, Teil 1 – Regelungsteil, o.O. 1997.
URL: http://www.informatik.uni-bremen.de/gdpa/z-pscrip.htm, Zugriff 18.03.2007.

Vögele, Hübner et al. (2003)
Vögele, T.; Hübner, S.; Schuster, G.: BUSTER – An Information Broker for the Semantic Web. In: KI – Künstliche Intelligenz, Jg. 17 (2003), Nr. 3, S. 31-34.

Vollmer (1994)
Vollmer, G.: Evolutionäre Erkenntnistheorie, 6. Aufl., Stuttgart 1994.

Vollrath (2000)
Vollrath, M.: Hier sitzt alles richtig – Die FMEA erfordert methodisches Know-how und Unterstützung seitens der Führung. In: QZ – Qualität und Zuverlässigkeit, Jg. 45 (2000), Nr. 12, S. 1567-1570.

von Ahsen, Lange (2004)
von Ahsen, A.; Lange, C.: Mehrdimensionale Fehlermöglichkeits- und -einflussanalyse als Instrument des Integrierten Qualitätsmanagements. In: ZfB, Jg. 74 (2004), Nr. 5, S. 441-460.

Vossen (1994)
Vossen, G.: Datenmodelle, Datenbanksprachen und Datenbank-Management-Systeme, 2. Aufl., Bonn 1994.

Wadsworth, Stephens et al. (2002)
Wadsworth, H. M.; Stephens, K. S.; Godfrey, A. B.: Modern Methods for Quality Control and Improvement, 2. Aufl., New York 2002.

Walder, Patzak (1997)
Walder, F.-P.; Patzak, G.: Qualitätsmanagement und Projektmanagement, Braunschweig 1997.

Wand (1996)
Wand, Y.: Ontology as a foundation for meta-modelling and method engineering. In: Information and Software Technology, Jg. 38 (1996), Nr. 4, S. 281-287.

Wand, Monarchi et al. (1995)
Wand, Y.; Monarchi, D. E.; Parsons, J.; Woo, C. C.: Theoretical foundations for conceptual modelling in information systems development. In: Decision Support Systems, Jg. 15 (1995), Nr. 4, S. 285-304.

Wand, Storey et al. (1999)
Wand, Y.; Storey, V. C.; Weber, R.: An Ontological Analysis of the Relationship Construct in Conceptual Modeling. In: ACM Transactions on Database Systems, Jg. 24 (1999), Nr. 4, S. 494-528.

Wand, Wang (1996)
Wand, Y.; Wang, R. Y.: Anchoring Data Quality Dimensions in Ontological Foundations. In: Communications of the ACM, Jg. 39 (1996), Nr. 11, S. 86-95.

Wand, Weber (2002)
Wand, Y.; Weber, R.: Research Commentary: Information Systems and Conceptual Modeling – A Research Agenda. In: Information Systems Research, Jg. 13 (2002), Nr. 4, S. 363-376.

Wang (2002)
Wang, Y.: Software Engineering Standards: Review and Perspectives. In: Chang, S. K. (Hrsg.): Handbook of Software Engineering und Knowledge Engineering, Vol. 1: Fundamentals, Singapur 2002, S. 277-304.

Warnecke, Kempf (1996)
Warnecke, H.-J.; Kempf, M.: Der Kausalprozessor. In: Pfeifer, T. (Hrsg.): Wissensbasierte Systeme in der Qualitätssicherung: Methoden zur Nutzung verteilten Wissens, Berlin 1996, S. 133-145.

Waterman (1986)
Waterman, D. A.: A Guide to Expert Systems. Reading 1986.

Weinrich (2003)
Weinrich, H.: Textgrammatik der deutschen Sprache, Mannheim 2003.

Weiss, Menzel et al. (2005)
Weiss, A.; Menzel, D.; Krohn, M.: Erfahrungswissen in industriellen Strukturen – Die Sicherung von Erfahrungswissen durch Transfer, Lernchancen und lernförderliche Arbeitsbedingungen. In: Meyer, J.-A. (Hrsg.): Wissens- und Informationsmanagement in kleinen und mittleren Unternehmen: Jahrbuch der KMU-Forschung und -Praxis 2005 in der Edition „Kleine und Mittlere Unternehmen“, Lohmar 2005, S. 127-141.

Weizenbaum (1977)
Weizenbaum, J.: Die Macht der Computer und die Ohnmacht der Vernunft. Frankfurt/Main 1977.

Welge, Al-Laham (2003)
Welge, M. K.; Al-Laham, A.: Strategisches Management – Grundlagen, Prozess, Implementierung, 4. Aufl., Wiesbaden 2003.

Wildemann (1998)
Wildemann, H.: FMEA – präventive Fehlervermeidung: Leitfaden zur Fehlerverbesserung und Fehlervermeidung für Konstruktions- und Gestaltungsprozesse, 5. Aufl., München 1998.

Wilhelm, Schorn (1999)
Wilhelm, D.; Schorn, M.: Über Fehler zu Wissenspotentialen – Das Fehler-Informations-System (FIS) als Baustein zum Wissensmanagement. In: QZ – Qualität und Zuverlässigkeit, Jg. 44 (1999), Nr. 3, S. 320-324.

Willke (2001)
Willke, H.: Systemisches Wissensmanagement, 2. Aufl., Stuttgart 2001.

Wirth, Berthold et al. (1996)
Wirth, R.; Berthold, B.; Krämer, A.; Peter, G.: Knowledge-based Support of System Analysis for the Analysis of Failure Modes and Effects. In: Engineering Applications of Artificial Intelligence, 9. Jg. (1996), Nr. 3, S. 219-229.

Witter (1995)
Witter, A.: Entwicklung eines Modells zur optimierten Nutzung des Wissenspotentials einer Prozess-FMEA, Dissertation Universität Hannover, Düsseldorf 1995.

Wolff (2004)
Wolff, M.: Abhandlung über die Prinzipien der Logik, Franfurt/Main 2004.

Woll, Zehl (2001)
Woll, R.; Zehl, R.: Transparente Informationen für Business Excellence. In: Redeker, G. (Hrsg.): Qualitätsmanagement für die Zukunft – Business Excellence als Ziel, Aachen 2001, S. 37-47.

Woll, Zehl et al. (2001)
Woll, R.; Zehl, R.; Krebs, I.: Quality Knowledge attained from Production for optimising the Development of Machinery. In: Weiss, Z. (Hrsg.): Manufacturing 2001, Wspólczesne Problemy Wytwarzania, Proceedings of the scientific-technical conference, Poznán 2001, S. 345-353.

Zehnder (1998)
Zehnder, C. A.: Informationssysteme und Datenbanken, 6. Aufl., Stuttgart 1998.

Zelewski (1995)
Zelewski, S.: Petrinetzbasierte Modellierung komplexer Produktionssysteme. Arbeitsbericht Nr. 7, Band 3, Universität Leipzig, Leipzig 1995.

Zelewski (2002a)
Zelewski, S.: Wissensmanagement mit Ontologien. In: Essener Unikate, Jg. 11 (2002), Nr. 18, S. 62-73.

Zelewski (2002b)
Zelewski, S.: Wissensmanagement mit Ontologien – eine einführende Darstellung. Arbeitsbericht Nr. 15. Universität Duisburg-Essen (Campus Essen), Essen 2002.

Zelewski (2005)
Zelewski, S.: Einführung in das Themenfeld „Ontologien“ aus informations- und betriebswirtschaftlicher Perspektive. In: Zelewski, S.; Alan, Y.; Alparslan, A.; Dittmann, L.; Weichelt, T. (Hrsg.): Ontologiebasierte Kompetenzmanagementsysteme – Grundlagen, Konzepte, Anwendungen, Berlin 2005, S. 115-228.

Zelewski, Alan et al. (2005)
Zelewski, S.; Alan, Y.; Alparslan, A.; Dittmann, L.; Weichelt, T. (Hrsg.): Ontologiebasierte Kompetenzmanagementsysteme – Grundlagen, Konzepte, Anwendungen, Berlin 2005.

Zelewski, Schütte et al. (2001)
Zelewski, S.; Schütte, R.; Siedentopf, J.: Ontologien zur Repräsentation von Domänen. In: Schreyögg, G. (Hrsg.): Wissen in Unternehmen – Konzepte, Maßnahmen, Methoden, Berlin 2001, S. 183-221.

Zenz (1999)
Zenz, A.: Strategisches Qualitätscontrolling – Konzeption als Metaführungsfunktion, Wiesbaden 1999.

Zollondz (2002)
Zollondz, H.-D.: Grundlagen Qualitätsmanagement: Einführung in Geschichte, Begriffe, Systeme und Konzepte, München 2002.

Anhang

A.1 Verwendete FMEA-Formulare der KSM nach VDA ´96

A.1.1 FMEA-Formular C 5960 A

Fehlermöglichkeits- und Einflussanalyse

FMEA-Nr.: C 5960 A

☑ **System-FMEA Produkt** ☐ **System-FMEA Prozess**

Seite 1 **von** 1

Typ/Modell/Fertigung/Charge: 9+1 Spindelschrauber	**Sach-Nr.:**	**Verantw.:** Müller	**Abt.:**
	Änderungsstand:	**Firma:** KSM	**Datum:** 03.06.2000
System-Nr./Systemelement: Bedienpult	**Sach-Nr.:**	**Verantw.:**	**Abt.:**
Funktion/Aufgabe: Umschaltung Automatik/Handbetrieb; schaltet Maschine ab (automatisch und manuell)	**Änderungsstand:**	**Firma:**	**Datum:**

Mögliche Fehlerfolge	B	Möglicher Fehler	Mögliche Fehlerursache	Vermeidungs-maßnahme	A	Entdeckungs-maßnahme	E	RPZ	V/T
Maschinenstörung (zu C 5960)	2	Anwahl Automatik und Handbetrieb nicht möglich	Pneumatikversorgung fehlt	---	1	Fehlermeldung (1) wurde programmiert	1	2	
	2		Druckschalter Pneumatik meldet nicht	---	1	Fehlermeldung (2) wurde programmiert	1	2	
	2		Schutztürschalter defekt	---	1	Fehlermeldung (3) wurde programmiert	1	2	
	2	Anwahl Automatik nicht möglich	Maschine (Atlas Copco) nicht betriebsbereit	---	1	Fehlermeldung (4) wurde programmiert	1	2	
	2		Maschine nicht in Grundstellung	---	1	Fehlermeldung (5) wurde programmiert	1	2	

B: Bedeutung des Fehlers
V/T: Verantwortlichkeit/Termin
Verantw.: Verantwortung
A: Wahrscheinlichkeit des Auftretens eines Fehlers
E: Wahrscheinlichkeit der Entdeckung eines Fehlers
Abt.: Abteilung
RPZ: Risikoprioritätszahl (B * A * E)

A.1.2 FMEA-Formular K 5965 A

Fehlermöglichkeits- und Einflussanalyse

FMEA-Nr.: K 5965 A

☑ **System-FMEA Produkt** ☐ **System-FMEA Prozess**

Seite 1 **von** 1

Typ/Modell/Fertigung/Charge: 5 Automatik Spindelschrauber	**Sach-Nr.:**	**Verantw.:** Müller	**Abt.:**
	Änderungsstand:	**Firma:** KSM	**Datum:** 03.06.2000
System-Nr./Systemelement: Bedienpult	**Sach-Nr.:**	**Verantw.:**	**Abt.:**
Funktion/Aufgabe: Umschaltung Automatik/Handbetrieb; schaltet Maschine ab (automatisch und manuell)	**Änderungsstand:**	**Firma:**	**Datum:**

Mögliche Fehlerfolge	B	Möglicher Fehler	Mögliche Fehlerursache	Vermeidungsmaßnahme	A	Entdeckungsmaßnahme	E	RPZ	V/T
Maschinenstörung (zu K 5965)	2	Anwahl Automatik und Handbetrieb nicht möglich	Pneumatikversorgung fehlt	---	1	Fehlermeldung (1) wurde programmiert	1	2	
	2		Hydraulikversorgung fehlt	---	1	Fehlermeldung (2) wurde programmiert	1	2	
			Mindest Füllstand Hydraulik unterschritten	Anfangsstand: 30.04.2000					
	2			---	3	Fehlermeldung (3) wurde programmiert	1	6	
				Änderungsstand: 03.06.2000					
	2			Wartungsplan überarbeiten (Öl nachfüllen)	1	Fehlermeldung (3) wurde programmiert	1	2	Firma Arndt
	2		Öltemperatur überschritten	---	1	Fehlermeldung (4) wurde programmiert	1	2	
	2		Druckschalter Hydraulik meldet nicht	---	1	Fehlermeldung (5) wurde programmiert	1	2	
			Ölfilter verschmutzt	Anfangsstand: 30.04.2000					
	2			---	4	Fehlermeldung (6) wurde programmiert	1	8	
				Änderungsstand: 03.06.2000					
	2			Wartungsplan überarbeiten (Filter austauschen)	1	Fehlermeldung (6) wurde programmiert	1	2	
	2		Druckschalter Pneumatik meldet nicht	---	1	Fehlermeldung (7) wurde programmiert	1	2	
	2		Schutztürschalter defekt	---	1	Fehlermeldung (8) wurde programmiert	1	2	
	2	Anwahl Automatik nicht möglich	Maschine (Atlas Copco) nicht betriebsbereit	---	1	Fehlermeldung (9) wurde programmiert	1	2	
	1		Maschine nicht in Grundstellung	---	1	Fehlermeldung (10) wurde programmiert	1	1	
Getriebe werden ohne Prozessdaten ausgeschleust		Fehlerhafter Werkstückträger wird bearbeitet	System zur Prozessdatenerfassung (Balogh) wurde abgestellt	Anfangsstand: 30.04.2000					
	3			---	3	---	5	45	
				Änderungsstand: 03.06.2000					
	3			---	3	System kann nur kontrolliert abgestellt werden	1	9	Meier
Getriebe muss in Reparaturschleife	3	Getriebe wird nicht bearbeitet	Zeitüberschreitung durch Rückstau	---	6	Fehlermeldung (11) wurde programmiert	1	18	

B: Bedeutung des Fehlers
V/T: Verantwortlichkeit/Termin
Verantw.: Verantwortung
A: Wahrscheinlichkeit des Auftretens eines Fehlers
E: Wahrscheinlichkeit der Entdeckung eines Fehlers
Abt.: Abteilung
RPZ: Risikoprioritätszahl (B * A * E)

A.1.3 FMEA-Formular K 5965 B

Fehlermöglichkeits- und Einflussanalyse

FMEA-Nr.: K 5965 B

☑ **System-FMEA Produkt** ☐ **System-FMEA Prozess**

Seite 1 **von** 2

Typ/Modell/Fertigung/Charge: 5 Automatik Spindelschrauber	**Sach-Nr.:**	**Verantw.:** Müller	**Abt.:**
	Änderungsstand:	**Firma:** KSM	**Datum:** 03.06.2000
System-Nr./Systemelement: Transportsystem im Maschinenbereich	**Sach-Nr.:**	**Verantw.:**	**Abt.:**
Funktion/Aufgabe: Transportiert Werkstückträger	**Änderungsstand:**	**Firma:**	**Datum:**

Mögliche Fehlerfolge	B	Möglicher Fehler	Mögliche Fehlerursache	Vermeidungs-maßnahme	A	Entdeckungs-maßnahme	E	RPZ	V/T
Produktionsausfall durch Not-Aus Betätigung (zu K 5965)	**2**	Konturbrille wurde von Getriebe betätigt	Getriebe ist in gekippter Stellung	---	**1**	Fehlermeldung (1) wurde programmiert	**1**	**2**	
Getriebe wird nicht bearbeitet (zu K 5965)	**3**	Datenformat nicht in Ordnung	Datenträger defekt	---	**2**	Fehlermeldung (2) wurde programmiert	**1**	**6**	
	3		Datenträger wurde nicht überschrieben	---	**2**	Fehlermeldung (3) wurde programmiert	**1**	**6**	
	3		Schreib-Lese-Kopf defekt	---	**1**	Fehlermeldung (4) wurde programmiert	**1**	**3**	
	3		Schreib-Lese-Kopf falsch eingerichtet	---	**2**	Fehlermeldung (5) wurde programmiert	**1**	**6**	
	3	Keine Verbindung zum System zur Produktionsdatenerfassung (Ballogh)	k. A.	---	**10**	---	**10**	**300**	
		Lichtschranke meldet nicht (Abfrage Bauzustand)	Keine Bauteile montiert	Anfangsstand: 30.04.2000					
	2			---	**4**	---	**4**	**32**	
				Änderungsstand: 03.06.2000					
	2			---	**4**	Fehlermeldung (6) wurde programmiert	**1**	**8**	Müller, Meier
	3		Lichtschranke defekt	---	**1**	Fehlermeldung (7) wurde programmiert	**1**	**3**	
	3		Lichtschranke falsch eingestellt	---	**1**	Fehlermeldung (8) wurde programmiert	**1**	**3**	
			Reflektor verschmutzt	Anfangsstand: 30.04.2000					
	3			---	**5**	Fehlermeldung (9) wurde programmiert	**1**	**15**	
				Änderungsstand: 03.06.2000					
	3			Wartungsplan überarbeitet	**1**	Fehlermeldung (9) wurde programmiert	**1**	**3**	Firma Arndt
			Reflektor defekt	Anfangsstand: 30.04.2000					
	3			---	**5**	Fehlermeldung (10) wurde programmiert	**1**	**15**	
				Änderungsstand: 03.06.2000					
	3			Wartungsplan überarbeitet	**1**	Fehlermeldung (10) wurde programmiert	**1**	**3**	Firma Arndt
	3	Betätigung der Taste „Getriebe nicht in Ordnung“	Taste gedrückt, obwohl Getriebe in Ordnung	---	**1**	---	**1**	**3**	
Produktionsausfall (zu K 5965)	**4**	Palettentransport gestört	Passfeder der Kupplung abgeschert	---	**1**	---	**1**	**4**	
	2		Friktion der Rollen zu schwach eingestellt	---	**1**	---	**1**	**2**	
	4		Kegelrad verschlissen	---	**1**	---	**1**	**4**	

FMEA-Nr.: K 5965B **System-Nr./Systemelement: Transportsystem im Maschinenbereich** **Seite 2 von 2**

Produktionsausfall (zu K 5965)	**3**	Vorstopper öffnet nicht	Schalter „Werkstückträger in Position (1)" defekt	---	**1**	Fehlermeldung (11) wurde programmiert	**1**	**3**
	3		Ventil (1) defekt	---	**1**	Fehlermeldung (12) wurde programmiert	**1**	**3**
	3		Ventilspule (1) defekt	---	**1**	Fehlermeldung (13) wurde programmiert	**1**	**3**
	2		Pneumatikversorgung fehlt (1)	---	**1**	Fehlermeldung (14) wurde programmiert	**1**	**2**
	2		Maschine ist nicht im Automatikbetrieb (1)	---	**1**	---	**1**	**2**
	2		Maschine ist abgeschaltet und Schlüsselschalter ist nicht auf Werkstückträger-Durchlauf gestellt (1)	---	**1**	---	**2**	**4**
	3	Maschinenstopper öffnet nicht	Schalter „Werkstückträger in Position (2)" defekt	---	**1**	Fehlermeldung (1) über Laufzeitkontrolle	**1**	**3**
	3		Ventil (2) defekt	---	**1**	Fehlermeldung (2) über Laufzeitkontrolle	**1**	**3**
	3		Ventilspule (2) defekt	---	**1**	Fehlermeldung (3) über Laufzeitkontrolle	**1**	**3**
	2		Pneumatikversorgung fehlt (2)	---	**1**	Fehlermeldung (15) wurde programmiert	**1**	**2**
	1		Maschine ist nicht im Automatikbetrieb (2)	---	**1**	Fehlermeldung (16) wurde programmiert	**1**	**1**
	2		Maschine ist abgeschaltet und Schlüsselschalter ist nicht auf Werkstückträger-Durchlauf gestellt (2)	---	**1**	---	**2**	**4**
Produktionsausfall (zu K 5965)	**2**	Werkstückträger wurde über den geschlossenen Vorstopper geschoben	Staudruck der Werkstückträger zu groß	---	**2**	---	**2**	**8**
	2	Zwei aneinanderstehende Werkstückträger am Maschinenstopper	Schutztür wurde geöffnet und der Schlüsselschalter Werkstückträger-Durchlauf stand auf öffnen	---	**3**	---	**2**	**12**
IO-Werkstückträger werden nicht bearbeitet (zu K 5965)	**3**	Werkstückträger werden unbearbeitet durchgeschleust	Ventil (3) defekt	---	**1**	Fehlermeldung (4) über Laufzeitkontrolle	**1**	**3**
	3		Ventilspule (3) defekt	---	**1**	Fehlermeldung (5) über Laufzeitkontrolle	**1**	**3**

B: Bedeutung des Fehlers — A: Wahrscheinlichkeit des Auftretens eines Fehlers — RPZ: Risikoprioritätszahl (B * A * E)
V/T: Verantwortlichkeit/Termin — E: Wahrscheinlichkeit der Entdeckung eines Fehlers
Verantw.: Verantwortung — Abt.: Abteilung

A.2 FMEA-Ontologie und Wissensbasis

A.2.1 FMEA-Ontologie mit Wissensbasis K 5965 A und K5965 B

```
// FMEA-Ontologie mit Instanziierung der Konzepte (Wissensbasis)

// Die folgenden zwei Zeilen legen die Namensräume der Ontologie fest, damit bei einer verteilten Anwen-
// dung eine eindeutige Referenzierung vorgenommen werden kann. Für herkömmliche "flache" Datentypen
// wird der offizielle Namensraum des W3-Konsortiums für XML genutzt und für alle OntoFMEA-spezifi-
// schen Konzepte wird als Namensraum „http://www.pim.uni-essen.de/ontofmea“ verwendet. Um Übersicht-
// lichkeit zu wahren, wird abkürzend für die jeweilige URL das Zeichen # als Platzhalter verwendet. Dabei
// wird zusätzlich für den Namensraum des W3-Konsortiums das Kürzel „xsd“ mit angeführt. Die verwendete
// Inferenzmaschine berücksichtigt dieses Vorgehen.
// Die abweichende Darstellung ermöglicht es, die vorliegende Kodierung ohne weitere Veränderungen mit
// der Inferenzmaschine verarbeiten zu können.
<ns
	ontons:xsd="http://www.w3.org/2001/XMLSchema"
	ontons="http://www.pim.uni-essen.de/ontofmea"
>

// ONTOLOGIE--------------------

// Konzepte --------------------

// Hauptkonzepte --------------------

#fmea::#DEFAULT_ROOT_CONCEPT.
#systemelement::#DEFAULT_ROOT_CONCEPT.
#funktion::#DEFAULT_ROOT_CONCEPT.
#fehler::#DEFAULT_ROOT_CONCEPT.
#fehlertupel::#DEFAULT_ROOT_CONCEPT.
#massnahme::#DEFAULT_ROOT_CONCEPT.
	#entdeckungsmassnahme::#massnahme.
	#vermeidungsmassnahme::#massnahme.
#datum::#DEFAULT_ROOT_CONCEPT.
#verantwortungstraeger::#DEFAULT_ROOT_CONCEPT.
	#unternehmen::#verantwortungstraeger.
	#person::#verantwortungstraeger.

// Struktur der Hauptkonzepte--------------------

#fmea[
	#ist_system_fmea_produkt=>xsd#BOOLEAN;
	#hat_fmea_nr=>xsd#STRING;
	#betrachtet_system_von_typ_modell_fertigung_charge=>xsd#STRING;
	#hat_verantwortung=>>#verantwortungstraeger;
	#wurde_erstellt=>#datum;
	#untersucht_systemelement=>#systemelement].

#systemelement[
	#hat_systemelementbezeichnung=>xsd#STRING;
	#ist_teil_von_systemelement=>>#systemelement;
	#ist_zusammengesetzt_aus_systemelement=>>#systemelement;
	#hat_schnittstelle_mit=>>#systemelement;
	#hat_funktion=>>#funktion;
	#wird_untersucht_von_fmea=>#fmea].
```

```
#funktion[
        #hat_funktionsbezeichnung=>xsd#STRING;
        #ist_funktion_von_systemelement=>>#systemelement;
        #ist_teil_von_funktion=>>#funktion;
        #ist_zusammengesetzt_aus_funktion=>>#funktion;
        #hat_fehlermoeglichkeit=>>#fehler].

#fehler[
        #hat_fehlerbezeichnung=>xsd#STRING;
        #verhindert_funktion=>>#funktion;
        #verursacht=>>#fehler;
        #hat_ursache=>>#fehler;
        #hat_gesamtrpz=>>xsd#INTEGER;
        #hat_fehlertupel=>>#fehlertupel].

#fehlertupel[
        #hat_bedeutung=>xsd#INTEGER;
        #hat_auftrittswahrscheinlichkeit=>xsd#INTEGER;
        #hat_entdeckungswahrscheinlichkeit=>xsd#INTEGER;
        #hat_rpz=>xsd#INTEGER;
        #ist_fehlerursache_zugeordnet=>>#fehler;
        #hat_massnahme=>>#massnahme;
        #ist_verantwortlich=>>#verantwortungstraeger;
        #hat_termin=>#datum;
        #datum_des_fehlertupels=>#datum].

#massnahme[
        #hat_bezeichnung=>xsd#STRING;
        #wird_angewendet=>>#fehlertupel].

#verantwortungstraeger[
        #hat_name=>xsd#STRING;
        #ist_verantwortlich_fuer_fehlertupel=>>fehlertupel].

#person[
        #arbeitet_fuer=>>#unternehmen].

#datum[
        #tag=>xsd#INTEGER;
        #monat=>xsd#INTEGER;
        #jahr=>xsd#INTEGER].

// Konzepte Funktionen und Funktionsstruktur ----------------------

#intransitive_funktion::#funktion.
        #abweichen::#intransitive_funktion.
        #arbeiten::#intransitive_funktion.
        #ausdehnen::#intransitive_funktion.
        #brennen::#intransitive_funktion.
        #einrasten::#intransitive_funktion.
        #fahren::#intransitive_funktion.
        #fallen::#intransitive_funktion.
        #federn::#intransitive_funktion.
        #erstarren::#intransitive_funktion.
        #fliessen::#intransitive_funktion.
        #folgen::#intransitive_funktion.
        #gleiten::#intransitive_funktion.
        #haften::#intransitive_funktion.
        #haengen::#intransitive_funktion.
        #laufen::#intransitive_funktion.
        #leuchten::#intransitive_funktion.
        #passen::#intransitive_funktion.
```

#pendeln::#intransitive_funktion.
#reagieren::#intransitive_funktion.
#rosten::#intransitive_funktion.
#schrumpfen::#intransitive_funktion.
#tasten::#intransitive_funktion.
#ueberlagern::#intransitive_funktion.
#wandeln::#intransitive_funktion.

#transitive_funktion::#funktion.
#positive_funktion::#transitive_funktion.
#abdecken::#positive_funktion.
#abgleichen::#positive_funktion.
#abgreifen::#positive_funktion.
#ablassen::#positive_funktion.
#abnehmen::#positive_funktion.
#abpressen::#positive_funktion.
#abschneiden::#positive_funktion.
#abschirmen::#positive_funktion.
#absetzen::#positive_funktion.
#absorbieren::#positive_funktion.
#abspritzen::#positive_funktion.
#abstimmen::#positive_funktion.
#abstreifen::#positive_funktion.
#abstufen::#positive_funktion.
#abtragen::#positive_funktion.
#abziehen::#positive_funktion.
#abzweigen::#positive_funktion.
#adsorbieren::#positive_funktion.
#analysieren::#positive_funktion.
#aendern::#positive_funktion.
#anpassen::#positive_funktion.
#anreichern::#positive_funktion.
#anstossen::#positive_funktion.
#antreiben::#positive_funktion.
#anzeigen::#positive_funktion.
#arretieren::#positive_funktion.
#aetzen::#positive_funktion.
#aufbereiten::#positive_funktion.
#aufgeben::#positive_funktion.
#aufnehmen::#positive_funktion.
#auftragen::#positive_funktion.
#aufzeichnen::#positive_funktion.
#ausbalancieren::#positive_funktion.
#ausbessern::#positive_funktion.
#ausloesen::#positive_funktion.
#ausrichten::#positive_funktion.
#ausschalten::#positive_funktion.
#bearbeiten::#positive_funktion.
#befestigen::#positive_funktion.
#beizen::#positive_funktion.
#beschaedigen::#negative_funktion.
#beschleunigen::#positive_funktion.
#beugen::#positive_funktion.
#bewerten::#positive_funktion.
#biegen::#positive_funktion.
#blockieren::#positive_funktion.
#bohren::#positive_funktion.
#brechen::#positive_funktion.
#bremsen::#positive_funktion.
#daempfen::#positive_funktion.
#demontieren::#positive_funktion.
#dehnen::#positive_funktion.

#dosieren::#positive_funktion.
#drehen::#positive_funktion.
#druecken::#positive_funktion.
#eichen::#positive_funktion.
#einfuegen::#positive_funktion.
#einschalten::#positive_funktion.
#emittieren::#positive_funktion.
#empfangen::#positive_funktion.
#formen::#positive_funktion.
#fraesen::#positive_funktion.
#fuehren::#positive_funktion.
#fuellen::#positive_funktion.
#giessen::#positive_funktion.
#gliedern::#positive_funktion.
#erzeugen::#positive_funktion.
#fertigen::#positive_funktion.
#fetten::#positive_funktion.
#filtern::#positive_funktion.
#fixieren::#positive_funktion.
#foerdern::#positive_funktion.
#greifen::#positive_funktion.
#halten::#positive_funktion.
#heben::#positive_funktion.
#heizen::#positive_funktion.
#hemmen::#positive_funktion.
#hobeln::#positive_funktion.
#impraegnieren::#positive_funktion.
#informieren::#positive_funktion.
#isolieren::#positive_funktion.
#justieren::#positive_funktion.
#kontrollieren::#positive_funktion.
#kopieren::#positive_funktion.
#koppeln::#positive_funktion.
#kuehlen::#positive_funktion.
#kuppeln::#positive_funktion.
#kalibrieren::#positive_funktion.
#kleben::#positive_funktion.
#kondensieren::#positive_funktion.
#lackieren::#positive_funktion.
#laden::#positive_funktion.
#lagern::#positive_funktion.
#leiten::#positive_funktion.
#lesen::#positive_funktion.
#loeschen::#positive_funktion.
#loesen::#positive_funktion.
#loeten::#positive_funktion.
#magnetisieren::#positive_funktion.
#markieren::#positive_funktion.
#messen::#positive_funktion.
#modulieren::#positive_funktion.
#montieren::#positive_funktion.
#neigen::#positive_funktion.
#nieten::#positive_funktion.
#nivellieren::#positive_funktion.
#nullen::#positive_funktion.
#polarisieren::#positive_funktion.
#polen::#positive_funktion.
#positionieren::#positive_funktion.
#praegen::#positive_funktion.
#pressen::#positive_funktion.
#pruefen::#positive_funktion.
#raffinieren::#positive_funktion.

#raeumen::#positive_funktion.
#rechnen::#positive_funktion.
#reduzieren::#positive_funktion.
#registrieren::#positive_funktion.
#reiben::#positive_funktion.
#reparieren::#positive_funktion.
#richten::#positive_funktion.
#saegen::#positive_funktion.
#sammeln::#positive_funktion.
#saugen::#positive_funktion.
#schalten::#positive_funktion.
#scheren::#positive_funktion.
#schieben::#positive_funktion.
#schleifen::#positive_funktion.
#schliessen::#positive_funktion.
#schmelzen::#positive_funktion.
#schmieden::#positive_funktion.
#schmieren::#positive_funktion.
#schneiden::#positive_funktion.
#schreiben::#positive_funktion.
#schuetteln::#positive_funktion.
#schuetten::#positive_funktion.
#schuetzen::#positive_funktion.
#schweissen::#positive_funktion.
#schwenken::#positive_funktion.
#setzen::#positive_funktion.
#sichern::#positive_funktion.
#sieben::#positive_funktion.
#sortieren::#positive_funktion.
#spalten::#positive_funktion.
#spannen::#positive_funktion.
#speichern::#positive_funktion.
#sperren::#positive_funktion.
#spuelen::#positive_funktion.
#stabilisieren::#positive_funktion.
#stanzen::#positive_funktion.
#stauchen::#positive_funktion.
#steuern::#positive_funktion.
#stoppen::#positive_funktion.
#stuetzen::#positive_funktion.
#tauschen::#positive_funktion.
#teilen::#positive_funktion.
#transportieren::#positive_funktion.
#trennen::#positive_funktion.
#trocknen::#positive_funktion.
#uebertragen::#positive_funktion.
#unterbrechen::#positive_funktion.
#verbinden::#positive_funktion.
#verdampfen::#positive_funktion.
#verdichten::#positive_funktion.
#verdunsten::#positive_funktion.
#vergleichen::#positive_funktion.
#versorgen::#positive_funktion.
#verteilen::#positive_funktion.
#verzoegern::#positive_funktion.
#walzen::#positive_funktion.
#waelzen::#positive_funktion.
#waermen::#positive_funktion.
#wechseln::#positive_funktion.
#wenden::#positive_funktion.
#wiegen::#positive_funktion.
#zaehlen::#positive_funktion.

```
                #zerlegen::#positive_funktion.
                #ziehen::#positive_funktion.
                #zufuegen::#positive_funktion.
                #zuenden::#positive_funktion.

        #negative_funktion::#transitive_funktion.
                #abnutzen::#negative_funktion.
                #angreifen::#negative_funktion.
                #beanspruchen::#negative_funktion.
                #belasten::#negative_funktion.
                #stoeren::#negative_funktion.
                #verschleissen::#negative_funktion.

        #neutrale_funktion::#transitive_funktion.
                #atomisieren::#neutrale_funktion.
                #ebnen::#neutrale_funktion.
                #erregen::#neutrale_funktion.
                #induzieren::#neutrale_funktion.
                #regeln::#neutrale_funktion.
                #rueckkoppeln::#neutrale_funktion.
                #skalieren::#neutrale_funktion.
                #transformieren::#neutrale_funktion.
                #umpolen::#neutrale_funktion.
                #umspannen::#neutrale_funktion.
                #zentrieren::#neutrale_funktion.


// Konzepte System und Systemstruktur

#system::#systemelement.
        #spanmaschine::#system.
        #schraubmaschine::#system.
        #schweissmaschine::#system.

einrichtung[ist_teil_von=>>system].
        gruppe[ist_teil_von=>>einrichtung].
                element[ist_teil_von=>>gruppe].
                        #verbindungselement::#element.
                                #metallfeder::#verbindungselement.
                                #gummifeder::#verbindungselement.
                                #gasfeder::#verbindungselement.
                        #elektrisches_element::#element.
                                #diode::#elektrisches_element.
                                #transistor::#elektrisches_element.
                                #thyristor::#elektrisches_element.
                                #passives_elektrisches_element::#elektrisches_element.
                                        #widerstand::#passives_elektrisches_element.
                                        #kondensator::#passives_elektrisches_element.
                                        #spule::#passives_elektrisches_element.
                                        #kabel::#passives_elektrisches_element.
                                        #sicherung::#passives_elektrisches_element.
                                #optoelektronisches_element::#elektrisches_element.
                                        #fotodiode::#optoelektronisches_element.
                                        #fototransistor::#optoelektronisches_element.
                                        #fotowiderstand::#optoelektronisches_element.
                                        #laserdiode::#optoelektronisches_element.
                                #elektrisches_regelelement::#elektrisches_element.
                                        #lichtschranke::#elektrisches_regelelement.
                                        #elektrisches_ventil::#elektrisches_regelelement.
                                        #elektrischer_schalter::#elektrisches_regelelement.
                                                #druckschalter::#elektrischer_schalter.
                                                #schutztuerschalter::#elektrischer_schalter.
```

```
                                        #schluesselschalter::#elektrischer_schalter.
                                        #zylinderschalter::#elektrischer_schalter.
                                #elektronisches_element::#elektrisches_element.
                                        #datentraeger::#elektronisches_element.
                                        #schreib_lese_kopf::#elektronisches_element.
                        #filterelement::#element.
                                #luftfilter::#filterelement.
                                #wasserfilter::#filterelement.
                                #oelfilter::#filterelement.
                        #antriebselement::#element.
                                #elektrischer_drehstrommotor::#antriebselement.
                                #elektrischer_gleichstrommotor::#antriebselement.
                        #transportelement::#element.
                                #gurt::#transportelement.
                                #rollen::#transportelement.
                                #kette::#transportelement.
                                #stopper::#transportelement.
                                #getriebeelement::#transportelement.
                        #gehaeuseelement::#element.
                                #rahmenelement::#gehaeuseelement.
                                #tuer::#gehaeuseelement.
                        #schrauberelement::#element.
                                #schrauberbit::#schrauberelement.
                                #schraubernuss::#schrauberelement.
                                #geber::#schrauberelement.
                                #schraubspindel::#schrauberelement.
                        #fuehrungselement::#element.
                                #linearfuehrung::#fuehrungselement.
                                #drehfuehrung::#fuehrungselement.
                        #hydraulisches_pneumatisches_element::#element.
                                #zylinder::#hydraulisches_pneumatisches_element.
                        #greifelement::#element.


#einheit::#systemelement.
        #baueinheit::#einheit.
        #betriebseinheit::#einheit.
        #funktionseinheit::#einheit.
                #bedieneinheit::#funktionseinheit.
                #feststelleinheit::#funktionseinheit.
                #transporteinheit::#funktionseinheit.
                #lagereinheit::#funktionseinheit.
                #demontageeinheit::#funktionseinheit.
                #hydraulikeinheit::#funktionseinheit.
                #pneumatikeinheit::#funktionseinheit.
                #gehaeuseeinheit::#funktionseinheit.
                #greifeinheit::#funktionseinheit.
                #werkstuecktraegerschwenkeinheit::#funktionseinheit.
                #schraubeinheit::#funktionseinheit.
                #schraubenvereinzelungseinheit::#funktionseinheit.
                #schraubensortiereinheit::#funktionseinheit.
                #prozessdatenerfassungseinheit::#funktionseinheit.
```

```
// WISSENSBASIS im engeren Sinne (Instanzen; Ausschnitte)--------------------

// FMEAS K 5965 --------------------

#k_5965_a_fmea_bedienpult:#fmea[
        #ist_system_fmea_produkt->"true";
        #hat_fmea_nr->"K 5965 A";
      #betrachtet_system_von_typ_modell_fertigung_charge->"5 Automatik Spindelschrauber-MB 80909";
        #hat_verantwortung->>#ksm_mueller;
        #wurde_erstellt->#_03062000;
        #untersucht_systemelement->#k_5965_a_bedienpult].


#k_5965_b_fmea_transportsystem:#fmea[
        #ist_system_fmea_produkt->"true";
        #hat_fmea_nr->"K 5965 B";
      #betrachtet_system_von_typ_modell_fertigung_charge->"5 Automatik Spindelschrauber-MB 80909";
        #hat_verantwortung->>#ksm_mueller;
        #wurde_erstellt->#_03062000;
        #untersucht_systemelement->#k_5965_b_transportsystem_im_maschinenbereich].


// SYSTEMELEMENTE K 5965 --------------------

#k_5965_fuenf_automatik_spindelschrauber:#schraubmaschine[
        #hat_systemelementbezeichnung->"5 Automatik Spindelschrauber";
        #ist_zusammengesetzt_aus_systemelement->>#k_5965_a_bedienpult;
        #ist_zusammengesetzt_aus_systemelement->>#k_5965_gehaeuse;
        #ist_zusammengesetzt_aus_systemelement->>#k_5965_hydraulikversorgung;
        #ist_zusammengesetzt_aus_systemelement->>#k_5965_pneumatikversorgung;
        #ist_zusammengesetzt_aus_systemelement->>#k_5965_prozessdatenerfassung;
        #ist_zusammengesetzt_aus_systemelement->>#k_5965_b_transportsystem_im_maschinenbereich;
        #ist_zusammengesetzt_aus_systemelement->>#k_5965_c_fixiereinheit;
        #ist_zusammengesetzt_aus_systemelement->>#k_5965_d_vertikalschlitten_des_schiebekamms;
        #ist_zusammengesetzt_aus_systemelement->>#k_5965_demontagehilfe;
#ist_zusammengesetzt_aus_systemelement->>#k_5965_e_horizontalschlitten_der_schraubenaufnahmeplatte;
        #ist_zusammengesetzt_aus_systemelement->>#k_5965_f_fuenf_spindel_schraubeinheit;
        #ist_zusammengesetzt_aus_systemelement->>#k_5965_g_schraubenvereinzelung;
        #ist_zusammengesetzt_aus_systemelement->>#k_5965_h_schwingsortierer;
        #ist_zusammengesetzt_aus_systemelement->>#k_5965_i_linearfoerderer;
        #ist_zusammengesetzt_aus_systemelement->>#k_5965_j_bunkerband;
        #ist_zusammengesetzt_aus_systemelement->>#k_5965_k_schraubenzufuehrband;
        #ist_zusammengesetzt_aus_systemelement->>#k_5965_bunker;
        #hat_funktion->>#k_5965_verschraubt_getriebe].

#k_5965_a_bedienpult:#bedieneinheit[
        #hat_systemelementbezeichnung->"Bedienpult";
        #ist_teil_von_systemelement->>#k_5965_fuenf_automatik_spindelschrauber;
        #hat_funktion->>#k_5965_a_umschaltung_automatik_manuell;
        #wird_untersucht_von_fmea->#k_5965_a_bedienpult].

#k_5965_a_taste_getriebe_nio:#elektrischer_schalter[
        #hat_systemelementbezeichnung->"Taste \"Getriebe nicht in Ordnung\" ";
        #ist_teil_von_systemelement->>#k_5965_a_bedienpult;
        #hat_funktion->>#k_5965_a_erlaubt_manuelle_statusaenderung].

#k_5965_a_schluesselschalter_wt_durchlauf:#schluesselschalter[
        #hat_systemelementbezeichnung->"Schlüsselschalter \"Werkstückträger Durchlauf\" ";
        #ist_teil_von_systemelement->>#k_5965_a_bedienpult;
        #hat_funktion->>#k_5965_a_gestattet_manuelles_einschalten_des_automatischen_durchlaufs].
```

```
#k_5965_gehaeuse:#gehaeuseeinheit[
    #hat_systemelementbezeichnung->"Gehäuse";
    #hat_funktion->>#k_5965_schuetzt_vor_ungesichertem_zugriff;
    #hat_funktion->>#k_5965_haelt_maschine_zusammen].

#k_5965_a_schutztuer:#tuer[
    #hat_systemelementbezeichnung->"Schutztür";
    #ist_teil_von_systemelement->>#k_5965_gehaeuse;
    #hat_funktion->>#k_5965_schuetzt_vor_zugriff_waehrend_des_betriebs].

#k_5965_konturbrille:#elektrischer_schalter[
    #hat_systemelementbezeichnung->"Konturbrille";
    #ist_teil_von_systemelement->>#k_5965_gehaeuse;
    #hat_funktion->>#k_5965_prueft_auf_richtige_stellung_des_getriebes;
   #hat_funktion->>#k_5965_schaltet_maschine_ab_sobald_mensch_kritischen_maschinenbereich_betritt].

#k_5965_schutztuerschalter:#schutztuerschalter[
    #hat_systemelementbezeichnung->"Schutztürschalter";
    #ist_teil_von_systemelement->>#k_5965_gehaeuse;
    #hat_funktion->>#k_5965_schutztuerschalter_schaltet_maschine_automatisch_ab].

#k_5965_hydraulikversorgung:#hydraulikeinheit[
    #hat_systemelementbezeichnung->"Hydraulikversorgung";
    #hat_funktion->>#k_5965_versorgt_maschine_mit_hydraulik].

#k_5965_druckschalter_hydraulik:#druckschalter[
    #hat_systemelementbezeichnung->"Druckschalter Hydraulik";
    #ist_teil_von_systemelement->>#k_5965_hydraulikversorgung;
    #hat_funktion->>#k_5965_druckschalter_hydraulik_schaltet_maschine_automatisch_ab].

#k_5965_oelfilter:#oelfilter[
    #hat_systemelementbezeichnung->"Ölfilter";
    #ist_teil_von_systemelement->>#k_5965_hydraulikversorgung;
    #hat_funktion->>#k_5965_filtert_oel_der_hydraulik].

#k_5965_pneumatikversorgung:#pneumatikeinheit[
    #hat_systemelementbezeichnung->"Pneumatikversorgung";
    #hat_funktion->>#k_5965_versorgt_maschine_mit_pneumatik].

#k_5965_druckschalter_pneumatik:#druckschalter[
    #hat_systemelementbezeichnung->"Druckschalter Pneumatik";
    #ist_teil_von_systemelement->>#k_5965_pneumatikversorgung;
    #hat_funktion->>#k_5965_druckschalter_pneumatik_schaltet_maschine_automatisch_ab].

#k_5965_prozessdatenerfassung:#prozessdatenerfassungseinheit[
    #hat_systemelementbezeichnung->"Prozessdatenerfassung (Balogh)";
    #hat_funktion->>#k_5965_erfasst_prozessdaten].

#k_5965_datentraeger:#datentraeger[
    #hat_systemelementbezeichnung->"Datenträger (montiert auf Werkstückträger)";
    #ist_teil_von_systemelement->>#k_5965_prozessdatenerfassung;
    #hat_funktion->>#k_5965_speichert_prozessdaten].

#k_5965_schreib_lese_kopf:#schreib_lese_kopf[
    #hat_systemelementbezeichnung->"Schreib-Lese-Kopf";
    #ist_teil_von_systemelement->>#k_5965_prozessdatenerfassung;
    #hat_funktion->>#k_5965_schreibt_liest_prozessdaten].

#k_5965_b_lichtschranke:#lichtschranke[
    #hat_systemelementbezeichnung->"Lichtschranke";
    #ist_teil_von_systemelement->>#k_5965_b_transportsystem_im_maschinenbereich;
    #hat_funktion->>#k_5965_b_prueft_bauzustand_des_getriebes].
```

```
#k_5965_b_transportsystem_im_maschinenbereich:#transporteinheit[
    #hat_systemelementbezeichnung->"Transportsystem im Maschinenbereich";
    #ist_teil_von_systemelement->>#k_5965_fuenf_automatik_spindelschrauber;
    #hat_funktion->>#k_5965_b_transportiert_werkstuecktraeger;
    #wird_untersucht_von_fmea->#k_5965_b_fmea_transportsystem].

#k_5965_b_passfeder:#metallfeder[
    #hat_systemelementbezeichnung->>"Passfeder";
    #ist_teil_von_systemelement->>#k_5965_b_transportsystem_im_maschinenbereich;
    #hat_funktion->>#k_5965_b_verbindet_kupplungselemente].


#k_5965_b_rollen:#rollen[
    #hat_systemelementbezeichnung->"Rollen";
    #ist_teil_von_systemelement->>#k_5965_b_transportsystem_im_maschinenbereich;
    #hat_funktion->>#k_5965_b_dient_als_gleitflaeche_fuer_wt].

#k_5965_b_kegelrad:#getriebeelement[
    #hat_systemelementbezeichnung->"Kegelrad";
    #ist_teil_von_systemelement->>#k_5965_b_transportsystem_im_maschinenbereich;
    #hat_funktion->>#k_5965_b_uebertraegt_kraft_an_rollen].

#k_5965_b_schalter_werkstueck_in_position_1:#elektrischer_schalter[
    #hat_systemelementbezeichnung->"Schalter \"Werkstück in Position\" (1)";
    #ist_teil_von_systemelement->>#k_5965_b_transportsystem_im_maschinenbereich;
    #hat_funktion->>#k_5965_b_gibt_signal_ueber_position_des_wts_1].

#k_5965_b_schalter_werkstueck_in_position_2:#elektrischer_schalter[
    #hat_systemelementbezeichnung->"Schalter \"Werkstück in Position\" (2)";
    #ist_teil_von_systemelement->>#k_5965_b_transportsystem_im_maschinenbereich;
    #hat_funktion->>#k_5965_b_gibt_signal_ueber_position_des_wts_2].

#k_5965_b_ventil_1:#elektrisches_ventil[
    #hat_systemelementbezeichnung->"Ventil (1)";
    #ist_teil_von_systemelement->>#k_5965_b_transportsystem_im_maschinenbereich;
    #hat_funktion->>#k_5965_b_befuellt_leert_zylinder_1].

#k_5965_b_ventilspule_1:#spule[
    #hat_systemelementbezeichnung->"Ventilspule (1)";
    #ist_teil_von_systemelement->>#k_5965_b_transportsystem_im_maschinenbereich;
    #hat_funktion->>#k_5965_b_oeffnet_schliesst_ventil_1].

#k_5965_b_ventil_2:#elektrisches_ventil[
    #hat_systemelementbezeichnung->"Ventil (2)";
    #ist_teil_von_systemelement->>#k_5965_b_transportsystem_im_maschinenbereich;
    #hat_funktion->>#k_5965_b_befuellt_leert_zylinder_2].

#k_5965_b_ventilspule_2:#spule[
    #hat_systemelementbezeichnung->"Ventilspule (2)";
    #ist_teil_von_systemelement->>#k_5965_b_transportsystem_im_maschinenbereich;
    #hat_funktion->>#k_5965_b_oeffnet_schliesst_ventil_2].

#k_5965_b_zylinder_1:#zylinder[
    #hat_systemelementbezeichnung->"Zylinder (1)";
    #ist_teil_von_systemelement->>#k_5965_b_transportsystem_im_maschinenbereich;
    #hat_funktion->>#k_5965_b_bewegt_vorstopper].

#k_5965_b_zylinder_2:#zylinder[
    #hat_systemelementbezeichnung->"Zylinder (2)";
    #ist_teil_von_systemelement->>#k_5965_b_transportsystem_im_maschinenbereich;
    #hat_funktion->>#k_5965_b_bewegt_maschinenstopper].
```

```
#k_5965_b_vorstopper:#stopper[
        #hat_systemelementbezeichnung->"Vorstopper";
        #ist_teil_von_systemelement->>#k_5965_b_transportsystem_im_maschinenbereich;
        #hat_funktion->>#k_5965_b_stoppt_wt_vor].

#k_5965_b_maschinenstopper:#stopper[
        #hat_systemelementbezeichnung->"Maschinenstopper";
        #ist_teil_von_systemelement->>#k_5965_b_transportsystem_im_maschinenbereich;
        #hat_funktion->>#k_5965_b_stoppt_wt].


// FUNKTIONEN K 5965 --------------------

#k_5965_verschraubt_getriebe:#fixieren[
        #hat_funktionsbezeichnung->"Verschraubt Getriebe";
        #ist_funktion_von_systemelement->>#k_5965_f_fuenf_spindel_verschraubt_getriebe;
        #ist_zusammengesetzt_aus_funktion->>#k_5965_b_transportiert_werkstuecktraeger;
        #ist_zusammengesetzt_aus_funktion->>#k_5965_f_fuenf_spindel_verschraubt_getriebe;
        #hat_fehlermoeglichkeit->>#k_5965_maschinenstoerung;
        #hat_fehlermoeglichkeit->>#k_5965_produktionsausfall;
        #hat_fehlermoeglichkeit->>#k_5965_getriebe_werden_ohne_prozessdaten_ausgeschleust;
        #hat_fehlermoeglichkeit->>#k_5965_getriebe_muss_in_reparaturschleife;
        #hat_fehlermoeglichkeit->>#k_5965_getriebe_wird_nicht_bearbeitet;
        #hat_fehlermoeglichkeit->>#k_5965_fehlerfreie_werkstuecktraeger_werden_nicht_bearbeitet;
        #hat_fehlermoeglichkeit->>#k_5965_moegliche_beschaedigung_von_bauteilen;
        #hat_fehlermoeglichkeit->>#k_5965_moegliche_beschaedigung_von_maschinenteilen;
        #hat_fehlermoeglichkeit->>#k_5965_drehmomente_und_winkelwerte_falsch;
        #hat_fehlermoeglichkeit->>#k_5965_h_sicherungseigenschaft_der_schrauben_laesst_nach].

#k_5965_a_umschaltung_automatik_manuell:#schalten[
        #hat_funktionsbezeichnung->"Umschaltung Automatik- oder Manuellbetrieb"].

#k_5965_a_erlaubt_manuelle_statusaenderung:#schalten[
        #hat_funktionsbezeichnung->"Erlaubt manuelle Statusänderung";
        #ist_teil_von_funktion->>#k_5965_a_umschaltung_automatik_manuell].

#k_5965_a_gestattet_manuelles_einschalten_des_automatischen_durchlaufs:#schalten[
        #hat_funktionsbezeichnung->"Gestattet manuelles Einschalten des automatischen Durchlaufs";
        #ist_teil_von_funktion->>#k_5965_a_umschaltung_automatik_manuell].

#k_5965_haelt_maschine_zusammen:#halten[
        #hat_funktionsbezeichnung->"Hält Maschine zusammen"].

#k_5965_schuetzt_vor_zugriff_waehrend_des_betriebs:#schuetzen[
        #hat_funktionsbezeichnung->"Schützt vor Zugriff während des Betriebs";
        #ist_teil_von_funktion->>#k_5965_haelt_maschine_zusammen].

#k_5965_schuetzt_vor_ungesichertem_zugriff:#schuetzen[
        #hat_funktionsbezeichnung->"Schützt vor ungesichertem Zugriff"].

#k_5965_prueft_auf_richtige_stellung_des_getriebes:#pruefen[
        #hat_funktionsbezeichnung->"Prüft auf richtige Stellung des Getriebes";
        #ist_teil_von_funktion->>#k_5965_schuetzt_vor_ungesichertem_zugriff].

#k_5965_schaltet_maschine_ab_sobald_mensch_kritischen_maschinenbereich_betritt:#schuetzen[
  #hat_funktionsbezeichnung->"Schaltet Maschine ab, sobald Mensch kritischen Maschinenbereich betritt";
        #ist_teil_von_funktion->>#k_5965_schuetzt_vor_ungesichertem_zugriff].

#k_5965_schutztuerschalter_schaltet_maschine_automatisch_ab:#schuetzen[
        #hat_funktionsbezeichnung->"Schutztürschalter schaltet Maschine automatisch ab";
        #ist_teil_von_funktion->>#k_5965_schuetzt_vor_ungesichertem_zugriff].
```

```
#k_5965_versorgt_maschine_mit_hydraulik:#versorgen[
        #hat_funktionsbezeichnung->"Versorgt Maschine mit Hydraulik"].

#k_5965_druckschalter_hydraulik_schaltet_maschine_automatisch_ab:#schalten[
        #hat_funktionsbezeichnung->"Druckschalter Hydraulik schaltet Maschine automatisch ab";
        #ist_teil_von_funktion->>#k_5965_versorgt_maschine_mit_hydraulik].

#k_5965_filtert_oel_der_hydraulik:#filtern[
        #hat_funktionsbezeichnung->"Filtert Öl der Hydraulik";
        #ist_teil_von_funktion->>#k_5965_versorgt_maschine_mit_hydraulik].

#k_5965_versorgt_maschine_mit_pneumatik:#versorgen[
        #hat_funktionsbezeichnung->"Versorgt Maschine mit Pneumatik"].

#k_5965_druckschalter_pneumatik_schaltet_maschine_automatisch_ab:#schalten[
        #hat_funktionsbezeichnung->"Druckschalter Pneumatik schaltet Maschine automatisch ab";
        #ist_teil_von_funktion->>#k_5965_versorgt_maschine_mit_pneumatik].

#k_5965_erfasst_prozessdaten:#aufzeichnen[
        #hat_funktionsbezeichnung->"Erfasst Prozessdaten"].

#k_5965_speichert_prozessdaten:#speichern[
        #hat_funktionsbezeichnung->"Speichert Prozessdaten";
        #ist_teil_von_funktion->>#k_5965_erfasst_prozessdaten].

#k_5965_schreibt_liest_prozessdaten:#speichern[
        #hat_funktionsbezeichnung->"Schreibt und liest Prozessdaten von bzw. auf den Datenträger";
        #ist_teil_von_funktion->>#k_5965_erfasst_prozessdaten].

#k_5965_b_transportiert_werkstuecktraeger:#transportieren[
        #hat_funktionsbezeichnung->"Transportiert Werkstückträger"].

#k_5965_b_prueft_bauzustand_des_getriebes:#pruefen[
        #hat_funktionsbezeichnung->"Prüft Bauzustand des Getriebes";
        #ist_teil_von_funktion->>#k_5965_b_transportiert_werkstuecktraeger].

#k_5965_b_verbindet_kupplungselemente:#verbinden[
        #hat_funktionsbezeichnung->"Verbindet Kupplungselemente";
        #ist_teil_von_funktion->>#k_5965_b_transportiert_werkstuecktraeger].

#k_5965_b_dient_als_gleitflaeche_fuer_werkstuecktraeger:#foerdern[
        #hat_funktionsbezeichnung->"Dient als Gleitfläche für Werkstückträger";
        #ist_teil_von_funktion->>#k_5965_b_transportiert_werkstuecktraeger].

#k_5965_b_uebertraegt_kraft_an_rollen:#uebertragen[
        #hat_funktionsbezeichnung->"Überträgt Kraft an Rollen";
        #ist_teil_von_funktion->>#k_5965_b_transportiert_werkstuecktraeger].

#k_5965_b_gibt_signal_ueber_position_des_werkstuecktraegers_1:#pruefen[
        #hat_funktionsbezeichnung->"Gibt Signal über die Position des Werkstückträgers (1)";
        #ist_teil_von_funktion->>#k_5965_b_transportiert_werkstuecktraeger].

#k_5965_b_gibt_signal_ueber_position_des_werkstuecktraegers_2:#pruefen[
        #hat_funktionsbezeichnung->"Gibt Signal über die Position des Werkstückträgers (2)";
        #ist_teil_von_funktion->>#k_5965_b_transportiert_werkstuecktraeger].

#k_5965_b_stoppt_werkstuecktraeger:#stoppen[
        #hat_funktionsbezeichnung->"Stoppt Werkstückträger";
        #ist_teil_von_funktion->>#k_5965_b_transportiert_werkstuecktraeger].
```

```
#k_5965_b_bewegt_maschinenstopper:#einschalten[
        #hat_funktionsbezeichnung->"Bewegt Maschinenstopper";
        #ist_teil_von_funktion->>#k_5965_b_stoppt_werkstuecktraeger].

#k_5965_b_befuellt_leert_zylinder_2:#schalten[
        #hat_funktionsbezeichnung->"Befüllt/Leert Zylinder (2)";
        #ist_teil_von_funktion->>#k_5965_b_bewegt_maschinenstopper].

#k_5965_b_stoppt_werkstuecktraeger_vor:#stoppen[
        #hat_funktionsbezeichnung->"Stoppt Werkstückträger vor";
        #ist_teil_von_funktion->>#k_5965_b_transportiert_werkstuecktraeger].

#k_5965_b_bewegt_vorstopper:#einschalten[
        #hat_funktionsbezeichnung->"Bewegt Vorstopper";
        #ist_teil_von_funktion->>#k_5965_b_stoppt_werkstuecktraeger_vor].

#k_5965_b_befuellt_leert_zylinder_1:#schalten[
        #hat_funktionsbezeichnung->"Befüllt/Leert Zylinder (1)";
        #ist_teil_von_funktion->>#k_5965_b_bewegt_vorstopper].

#k_5965_b_oeffnet_schliesst_ventil_1:#schalten[
        #hat_funktionsbezeichnung->"Öffnet/Schließt Ventil (1)";
        #ist_teil_von_funktion->>#k_5965_b_befuellt_leert_zylinder_1].

#k_5965_b_oeffnet_schliesst_ventil_2:#schalten[
        #hat_funktionsbezeichnung->"Öffnet/Schließt Ventil (2)";
        #ist_teil_von_funktion->>#k_5965_b_befuellt_leert_zylinder_2].


// FEHLER K 5965 --------------------

#k_5965_produktionsausfall:#fehler[
                #hat_fehlerbezeichnung->"Produktionsausfall"].

#k_5965_maschinenstoerung:#fehler[
                #hat_fehlerbezeichnung->"Maschinenstörung";
                #verursacht->>#k_5965_produktionsausfall].

#k_5965_getriebe_werden_ohne_prozessdaten_ausgeschleust:#fehler[
                #hat_fehlerbezeichnung->"Getriebe werden ohne Prozessdaten ausgeschleust"].

#k_5965_getriebe_muss_in_reparaturschleife:#fehler[
                #hat_fehlerbezeichnung->"Getriebe muss in Reparaturschleife"].

#k_5965_getriebe_wird_nicht_bearbeitet:#fehler[
                #hat_fehlerbezeichnung->"Getriebe wird nicht bearbeitet"].

#k_5965_fehlerfreie_werkstuecktraeger_werden_nicht_bearbeitet:#fehler[
               #hat_fehlerbezeichnung->"Fehlerfreie Werkstückträger (IO WTs) werden nicht bearbeitet"].

#k_5965_moegliche_beschaedigung_von_bauteilen:#fehler[
                #hat_fehlerbezeichnung->"Mögliche Beschädigung von Bauteilen";
                #verursacht->>#k_5965_maschinenstoerung].

#k_5965_moegliche_beschaedigung_von_maschinenteilen:#fehler[
                #hat_fehlerbezeichnung->"Mögliche Beschädigung von Maschinenteilen";
                #verursacht->>#k_5965_maschinenstoerung].

#k_5965_drehmomente_und_winkelwerte_falsch:#fehler[
                #hat_fehlerbezeichnung->"Drehmomente und Winkelwerte falsch"].
```

```
#k_5965_a_anwahl_automatik_und_handbetrieb_nicht_moeglich:#fehler[
            #hat_fehlerbezeichnung->"Anwahl Automatik und Handbetrieb nicht möglich";
            #verhindert_funktion->>#k_5965_a_umschaltung_automatik_manuell;
            #verursacht->>#k_5965_maschinenstoerung].

#k_5965_a_anwahl_automatik_nicht_moeglich:#fehler[
            #hat_fehlerbezeichnung->"Anwahl Automatik nicht möglich";
            #verhindert_funktion->>#k_5965_a_umschaltung_automatik_manuell;
            #verursacht->>#k_5965_maschinenstoerung].

#k_5965_a_fehlerhafter_werkstuecktraeger_wird_bearbeitet:#fehler[
            #hat_fehlerbezeichnung->"Fehlerhafter Werkstückträger wird bearbeitet";
            #verursacht->>#k_5965_getriebe_werden_ohne_prozessdaten_ausgeschleust].
#k_5965_a_getriebe_wird_nicht_bearbeitet:#fehler[
            #hat_fehlerbezeichnung->"Getriebe wird nicht bearbeitet";
            #verursacht->>#k_5965_getriebe_muss_in_reparaturschleife].

#k_5965_a_pneumatikversorgung_fehlt:#fehler[
            #hat_fehlerbezeichnung->"Pneumatikversorgung fehlt";
            #verhindert_funktion->>#k_5965_versorgt_maschine_mit_pneumatik;
            #verursacht->>#k_5965_a_anwahl_automatik_und_handbetrieb_nicht_moeglich].

#k_5965_a_hydraulikversorgung_fehlt:#fehler[
            #hat_fehlerbezeichnung->"Hydraulikversorgung fehlt";
            #verhindert_funktion->>#k_5965_versorgt_maschine_mit_hydraulik;
            #verursacht->>#k_5965_a_anwahl_automatik_und_handbetrieb_nicht_moeglich].

#k_5965_a_mindestfuellstand_hydraulik_unterschritten:#fehler[
            #hat_fehlerbezeichnung->"Mindestfüllstand Hydraulik unterschritten";
            #verhindert_funktion->>#k_5965_versorgt_maschine_mit_hydraulik;
            #verursacht->>#k_5965_a_anwahl_automatik_und_handbetrieb_nicht_moeglich].

#k_5965_a_oeltemperatur_ueberschritten:#fehler[
            #hat_fehlerbezeichnung->"Öltemperatur überschritten";
            #verhindert_funktion->>#k_5965_versorgt_maschine_mit_hydraulik;
            #verursacht->>#k_5965_a_anwahl_automatik_und_handbetrieb_nicht_moeglich].

#k_5965_a_druckschalter_hydraulik_meldet_nicht:#fehler[
            #hat_fehlerbezeichnung->"Druckschalter Hydraulik meldet nicht";
        #verhindert_funktion->>#k_5965_druckschalter_hydraulik_schaltet_maschine_automatisch_ab;
            #verursacht->>#k_5965_a_anwahl_automatik_und_handbetrieb_nicht_moeglich].

#k_5965_a_oelfilter_verschmutzt:#fehler[
            #hat_fehlerbezeichnung->"Ölfilter verschmutzt";
            #verhindert_funktion->>#k_5965_filtert_oel_der_hydraulik;
            #verursacht->>#k_5965_a_anwahl_automatik_und_handbetrieb_nicht_moeglich].

#k_5965_a_druckschalter_pneumatik_meldet_nicht:#fehler[
            #hat_fehlerbezeichnung->"Druckschalter Pneumatik meldet nicht";
        #verhindert_funktion->>#k_5965_druckschalter_pneumatik_schaltet_maschine_automatisch_ab;
            #verursacht->>#k_5965_a_anwahl_automatik_und_handbetrieb_nicht_moeglich].

#k_5965_a_schutztuerschalter_defekt:#fehler[
            #hat_fehlerbezeichnung->"Schutztürschalter defekt";
            #verhindert_funktion->>#k_5965_schutztuerschalter_schaltet_maschine_automatisch_ab;
            #verursacht->>#k_5965_a_anwahl_automatik_und_handbetrieb_nicht_moeglich].

#k_5965_a_maschine_nicht_betriebsbereit:#fehler[
            #hat_fehlerbezeichnung->"Schrauber (Atlas Copco) nicht betriebsbereit";
            #verhindert_funktion->>#k_5965_f_schraubt_schrauben;
            #verursacht->>#k_5965_a_anwahl_automatik_nicht_moeglich].
```

```
#k_5965_a_maschine_nicht_in_grundstellung:#fehler[
                #hat_fehlerbezeichnung->"Maschine nicht in Grundstellung";
                #verursacht->>#k_5965_a_anwahl_automatik_nicht_moeglich].

#k_5965_a_system_zur_pde_wurde_abgestellt:#fehler[
                #hat_fehlerbezeichnung->"System zur Prozessdatenerfassung (Balogh) wurde abgestellt";
                #verhindert_funktion->>#k_5965_erfasst_prozessdaten;
                #verursacht->>#k_5965_a_fehlerhafter_werkstuecktraeger_wird_bearbeitet].

#k_5965_a_zeitueberschreitung_durch_rueckstau:#fehler[
                #hat_fehlerbezeichnung->"Zeitüberschreitung durch Rückstau";
                #verursacht->>#k_5965_a_getriebe_wird_nicht_bearbeitet].

#k_5965_b_konturbrille_wurde_betaetigt:#fehler[
                #hat_fehlerbezeichnung->"Konturbrille wurde von Getriebe betätigt";
                #verhindert_funktion->>#k_5965_prueft_auf_richtige_stellung_des_getriebes;
                #verhindert_funktion->>#k_5965_schaltet_maschine_ab_sobald_mensch_kritischen_ma-
schinenbereich_betritt;
                #verursacht->>#k_5965_produktionsausfall].

#k_5965_b_datenformat_1_nicht_in_ordnung:#fehler[
                #hat_fehlerbezeichnung->"Datenformat nicht in Ordnung";
                #verhindert_funktion->>#k_5965_erfasst_prozessdaten;
                #verursacht->>#k_5965_getriebe_wird_nicht_bearbeitet;
                #verursacht->>#k_5965_getriebe_muss_in_reparaturschleife].

#k_5965_b_keine_verbindung_zum_system_zur_pde_1:#fehler[
           #hat_fehlerbezeichnung->"Keine Verbindung zum System zur Prozessdatenerfassung (Balogh)";
                #verhindert_funktion->>#k_5965_erfasst_prozessdaten;
                #verursacht->>#k_5965_getriebe_wird_nicht_bearbeitet].

#k_5965_b_lichtschranke_meldet_nicht:#fehler[
                #hat_fehlerbezeichnung->"Lichtschranke meldet nicht (Abfrage Bauzustand)";
                #verhindert_funktion->>#k_5965_b_prueft_bauzustand_des_getriebes;
                #verursacht->>#k_5965_getriebe_wird_nicht_bearbeitet].

#k_5965_a_betaetigung_der_taste_getriebe_nicht_in_ordnung:#fehler[
                #hat_fehlerbezeichnung->"Betätigung der Taste \"Getriebe nicht in Ordnung\"";
                #verhindert_funktion->>#k_5965_a_erlaubt_manuelle_statusaenderung;
                #verursacht->>#k_5965_getriebe_wird_nicht_bearbeitet].

#k_5965_b_palettentransport_gestoert:#fehler[
                #hat_fehlerbezeichnung->"Transport des Werkstückträgers gestört";
                #verhindert_funktion->>#k_5965_b_transportiert_werkstuecktraeger;
                #verursacht->>#k_5965_produktionsausfall].

#k_5965_b_getriebe_ist_in_gekippter_stellung:#fehler[
                #hat_fehlerbezeichnung->"Getriebe ist in gekippter Stellung";
                #verhindert_funktion->>#k_5965_prueft_auf_richtige_stellung_des_getriebes;
                #verursacht->>#k_5965_b_konturbrille_wurde_betaetigt].

#k_5965_b_datentraeger_1_defekt:#fehler[
                #hat_fehlerbezeichnung->"Datenträger defekt";
                #verhindert_funktion->>#k_5965_speichert_prozessdaten;
                #verursacht->>#k_5965_b_datenformat_1_nicht_in_ordnung].

#k_5965_b_datentraeger_1_wurde_nicht_ueberschrieben:#fehler[
                #hat_fehlerbezeichnung->"Datenträger wurde nicht überschrieben";
                #verhindert_funktion->>#k_5965_speichert_prozessdaten;
                #verursacht->>#k_5965_b_datenformat_1_nicht_in_ordnung].
```

```
#k_5965_b_schreib_lese_kopf_1_defekt:#fehler[
                    #hat_fehlerbezeichnung->"Schreib-Lese-Kopf defekt";
                    #verhindert_funktion->>#k_5965_schreibt_liest_prozessdaten;
                    #verursacht->>#k_5965_b_datenformat_1_nicht_in_ordnung].

#k_5965_b_schreib_lese_kopf_1_falsch_eingerichtet:#fehler[
                    #hat_fehlerbezeichnung->"Schreib-Lese-Kopf falsch eingerichtet";
                    #verhindert_funktion->>#k_5965_schreibt_liest_prozessdaten;
                    #verursacht->>#k_5965_b_datenformat_1_nicht_in_ordnung].

#k_5965_b_keine_angabe_1:#fehler[
                    #hat_fehlerbezeichnung->"k. A.";
                    #verursacht->>#k_5965_b_keine_verbindung_zum_system_zur_pde_1].

#k_5965_b_keine_angabe_2:#fehler[
                    #hat_fehlerbezeichnung->"k. A.";
                    #verursacht->>#k_5965_b_keine_verbindung_zum_system_zur_pde_2].

#k_5965_b_keine_bauteile_montiert:#fehler[
                    #hat_fehlerbezeichnung->"Keine Bauteile montiert";
                    #verursacht->>#k_5965_b_lichtschranke_meldet_nicht].

#k_5965_b_lichtschranke_defekt:#fehler[
                    #hat_fehlerbezeichnung->"Lichtschranke defekt";
                    #verhindert_funktion->>#k_5965_b_prueft_bauzustand_des_getriebes;
                    #verursacht->>#k_5965_b_lichtschranke_meldet_nicht].

#k_5965_b_lichtschranke_falsch_eingestellt:#fehler[
                    #hat_fehlerbezeichnung->"Lichtschranke falsch eingestellt";
                    #verhindert_funktion->>#k_5965_b_prueft_bauzustand_des_getriebes;
                    #verursacht->>#k_5965_b_lichtschranke_meldet_nicht].

#k_5965_b_reflektor_verschmutzt:#fehler[
                    #hat_fehlerbezeichnung->"Reflektor verschmutzt";
                    #verhindert_funktion->>#k_5965_b_prueft_bauzustand_des_getriebes;
                    #verursacht->>#k_5965_b_lichtschranke_meldet_nicht].

#k_5965_b_reflektor_defekt:#fehler[
                    #hat_fehlerbezeichnung->"Reflektor defekt";
                    #verhindert_funktion->>#k_5965_b_prueft_bauzustand_des_getriebes;
                    #verursacht->>#k_5965_b_lichtschranke_meldet_nicht].

#k_5965_a_taste_getriebe_nicht_in_ordnung_gedrueckt:#fehler[
     #hat_fehlerbezeichnung->"Taste \"Getriebe nicht in Ordnung\" gedrückt, obwohl Getriebe in Ordnung";
                    #verhindert_funktion->>#k_5965_a_erlaubt_manuelle_statusaenderung;
                    #verursacht->>#k_5965_b_betaetigung_der_taste_getriebe_nicht_in_ordnung].

#k_5965_b_passfeder_der_kupplung_abgeschert:#fehler[
                    #hat_fehlerbezeichnung->"Passfeder der Kupplung abgeschert";
                    #verhindert_funktion->>#k_5965_b_verbindet_kupplungselemente;
                    #verursacht->>#k_5965_b_palettentransport_gestoert].

#k_5965_b_friktion_der_rollen_zu_schwach_eingestellt:#fehler[
                    #hat_fehlerbezeichnung->"Friktion der Rollen zu schwach eingestellt";
                    #verhindert_funktion->>#k_5965_b_dient_als_gleitflaeche_fuer_wt;
                    #verursacht->>#k_5965_b_palettentransport_gestoert].

#k_5965_b_kegelrad_verschlissen:#fehler[
                    #hat_fehlerbezeichnung->"Kegelrad verschlissen";
                    #verhindert_funktion->>#k_5965_b_uebertraegt_kraft_an_rollen;
                    #verursacht->>#k_5965_b_palettentransport_gestoert].
```

#k_5965_b_vorstopper_oeffnet_nicht:#fehler[
#hat_fehlerbezeichnung->"Vorstopper öffnet nicht";
#verhindert_funktion->>#k_5965_b_stoppt_wt_vor;
#verursacht->>#k_5965_produktionsausfall].

#k_5965_b_maschinenstopper_oeffnet_nicht:#fehler[
#hat_fehlerbezeichnung->"Maschinenstopper öffnet nicht";
#verhindert_funktion->>#k_5965_b_stoppt_wt;
#verursacht->>#k_5965_produktionsausfall].

#k_5965_b_wertstuecktraeger_wurde_ueber_geschlossenen_vorstopper:#fehler[
#hat_fehlerbezeichnung->"Werkstückträger wurde über geschlossenen Vorstopper geschoben";
#verursacht->>#k_5965_produktionsausfall].

#k_5965_b_zwei_aneinanderstehende_werkstuecktraeger_stehen:#fehler[
#hat_fehlerbezeichnung->"Zwei aneinanderstehende Werkstückträger stehen am Maschinenstopper";
#verhindert_funktion->>#k_5965_b_stoppt_wt;
#verursacht->>#k_5965_produktionsausfall].

#k_5965_b_werkstuecktraeger_werden_unbearbeitet_durchgeschleust:#fehler[
#hat_fehlerbezeichnung->"Werkstückträger werden unbearbeitet durchgeschleust";
#verhindert_funktion->>#k_5965_b_stoppt_wt;
#verursacht->>#k_5965_fehlerfreie_werkstuecktraeger_werden_nicht_bearbeitet].

#k_5965_b_schalter_werkstuecktraeger_in_position_1_defekt:#fehler[
#hat_fehlerbezeichnung->"Schalter \"Werkstückträger in Position\" (1) defekt";
#verhindert_funktion->>#k_5965_b_gibt_signal_ueber_position_des_wts_1;
#verursacht->>#k_5965_b_vorstopper_oeffnet_nicht].

#k_5965_b_ventil_1_defekt:#fehler[
#hat_fehlerbezeichnung->"Ventil (1) defekt";
#verhindert_funktion->>#k_5965_b_befuellt_leert_zylinder_1;
#verursacht->>#k_5965_b_vorstopper_oeffnet_nicht].

#k_5965_b_ventilspule_1_defekt:#fehler[
#hat_fehlerbezeichnung->"Ventilspule (1) defekt";
#verhindert_funktion->>#k_5965_b_oeffnet_schliesst_ventil_1;
#verursacht->>#k_5965_b_vorstopper_oeffnet_nicht].

#k_5965_b_maschine_1_nicht_im_automatikbetrieb:#fehler[
#hat_fehlerbezeichnung->"Maschine nicht im Automatikbetrieb";
#verursacht->>#k_5965_b_vorstopper_oeffnet_nicht].

#k_5965_b_maschine_1_ist_abgeschaltet_und_schluesselschalter:#fehler[
#hat_fehlerbezeichnung->"Maschine ist abgeschaltet und Schlüsselschalter nicht auf Werkstückträger-Durchlauf gestellt";
#verursacht->>#k_5965_b_vorstopper_oeffnet_nicht].

#k_5965_b_schalter_werkstuecktraeger_in_position_2_defekt:#fehler[
#hat_fehlerbezeichnung->"Schalter \"Werkstückträger in Position\" (2) defekt";
#verhindert_funktion->>#k_5965_b_gibt_signal_ueber_position_des_wts_2;
#verursacht->>#k_5965_b_maschinenstopper_oeffnet_nicht].

#k_5965_b_ventil_2_defekt:#fehler[
#hat_fehlerbezeichnung->"Ventil (2) defekt";
#verhindert_funktion->>#k_5965_b_befuellt_leert_zylinder_2;
#verursacht->>#k_5965_b_maschinenstopper_oeffnet_nicht].

#k_5965_b_ventilspule_2_defekt:#fehler[
#hat_fehlerbezeichnung->"Ventilspule (2) defekt";
#verhindert_funktion->>#k_5965_b_oeffnet_schliesst_ventil_2;
#verursacht->>#k_5965_b_maschinenstopper_oeffnet_nicht].

#k_5965_b_maschine_2_nicht_im_automatikbetrieb:#fehler[
#hat_fehlerbezeichnung->"Maschine nicht im Automatikbetrieb";
#verursacht->>#k_5965_b_maschinenstopper_oeffnet_nicht].

#k_5965_b_maschine_2_ist_abgeschaltet_und_schluesselschalter:#fehler[
#hat_fehlerbezeichnung->"Maschine ist abgeschaltet und Schlüsselschalter ist auf Werkstückträger-Durchlauf gestellt";
#verursacht->>#k_5965_b_maschinenstopper_oeffnet_nicht].

#k_5965_b_staudruck_der_werkstuecktraeger_zu_gross:#fehler[
#hat_fehlerbezeichnung->"Staudruck der Werkstückträger zu groß";
#verhindert_funktion->>#k_5965_b_stoppt_wt_vor;
#verursacht->>#k_5965_b_wertstuecktraeger_wurde_ueber_geschlossenen_vorstopper].
#k_5965_b_schutztuer_wurde_geoeffnet_und_schluesselschalter:#fehler[
#hat_fehlerbezeichnung->"Schutztür wurde geöffnet und Schlüsselschalter Werkstückträger-Durchlauf stand auf öffnen";
#verursacht->>#k_5965_b_zwei_aneinanderstehende_werkstuecktraeger_stehen].

#k_5965_b_ventil_3_defekt:#fehler[
#hat_fehlerbezeichnung->"Ventil (3) defekt";
#verursacht->>#k_5965_b_werkstuecktraeger_werden_unbearbeitet_durchgeschleust].

#k_5965_b_ventilspule_3_defekt:#fehler[
#hat_fehlerbezeichnung->"Ventilspule (3) defekt";
#verursacht->>#k_5965_b_werkstuecktraeger_werden_unbearbeitet_durchgeschleust].

#k_5965_b_pneumatikversorgung_1_fehlt:#fehler[
#hat_fehlerbezeichnung->"Pneumatikversorgung fehlt";
#verhindert_funktion->>#k_5965_versorgt_maschine_mit_pneumatik;
#verursacht->>#k_5965_b_vorstopper_oeffnet_nicht].

#k_5965_b_pneumatikversorgung_2_fehlt:#fehler[
#hat_fehlerbezeichnung->"Pneumatikversorgung fehlt";
#verhindert_funktion->>#k_5965_b_versorgt_maschine_mit_pneumatik;
#verursacht->>#k_5965_b_maschinenstopper_oeffnet_nicht].

// FEHLERTUPEL K 5965 --------------------

#k_5965_a_fehlertupel_pneumatikversorgung_fehlt:#fehlertupel[
#hat_bedeutung->2.0;
#hat_auftrittswahrscheinlichkeit->1.0;
#hat_entdeckungswahrscheinlichkeit->1.0;
#ist_fehlerursache_zugeordnet->>#k_5965_a_pneumatikversorgung_fehlt;
#hat_massnahme->>#k_5965_a_fehlermeldung_1_wurde_programmiert].

#k_5965_a_fehlertupel_hydraulikversorgung_fehlt:#fehlertupel[
#hat_bedeutung->2.0;
#hat_auftrittswahrscheinlichkeit->1.0;
#hat_entdeckungswahrscheinlichkeit->1.0;
#ist_fehlerursache_zugeordnet->>#k_5965_a_hydraulikversorgung_fehlt;
#hat_massnahme->>#k_5965_a_fehlermeldung_2_wurde_programmiert].

#k_5965_a_fehlertupel_mindestfuellstand_unterschritten_1:#fehlertupel[
#hat_bedeutung->2.0;
#hat_auftrittswahrscheinlichkeit->3.0;
#hat_entdeckungswahrscheinlichkeit->1.0;
#ist_fehlerursache_zugeordnet->>#k_5965_a_mindestfuellstand_hydraulik_unterschritten;
#hat_massnahme->>#k_5965_a_fehlermeldung_3_wurde_programmiert;
#datum_des_fehlertupels->#_30042000].

```
#k_5965_a_fehlertupel_mindestfuellstand_unterschritten_2:#fehlertupel[
        #hat_bedeutung->2.0;
        #hat_auftrittswahrscheinlichkeit->1.0;
        #hat_entdeckungswahrscheinlichkeit->1.0;
        #ist_fehlerursache_zugeordnet->>#k_5965_a_mindestfuellstand_hydraulik_unterschritten;
        #hat_massnahme->>#k_5965_a_fehlermeldung_3_wurde_programmiert;
        #hat_massnahme->>#k_5965_a_wartungsplan_ueberarbeitet_1;
        #ist_verantwortlich->>#arndt;
        #datum_des_fehlertupels->#_03062000].

#k_5965_a_fehlertupel_oeltemperatur_ueberschritten:#fehlertupel[
        #hat_auftrittswahrscheinlichkeit->1.0;
        #hat_bedeutung->2.0;
        #hat_entdeckungswahrscheinlichkeit->1.0;
        #ist_fehlerursache_zugeordnet->>#k_5965_a_oeltemperatur_ueberschritten;
        #hat_massnahme->>#k_5965_a_fehlermeldung_4_wurde_programmiert].

#k_5965_a_fehlertupel_druckschalter_hydraulik_meldet_nicht:#fehlertupel[
        #hat_auftrittswahrscheinlichkeit->1.0;
        #hat_bedeutung->2.0;
        #hat_entdeckungswahrscheinlichkeit->1.0;
        #ist_fehlerursache_zugeordnet->>#k_5965_a_druckschalter_hydraulik_meldet_nicht;
        #hat_massnahme->>#k_5965_a_fehlermeldung_5_wurde_programmiert].

#k_5965_a_fehlertupel_oelfilter_verschmutzt_1:#fehlertupel[
        #hat_bedeutung->2.0;
        #hat_auftrittswahrscheinlichkeit->4.0;
        #hat_entdeckungswahrscheinlichkeit->1.0;
        #ist_fehlerursache_zugeordnet->>#k_5965_a_oelfilter_verschmutzt;
        #hat_massnahme->>#k_5965_a_fehlermeldung_6_wurde_programmiert;
        #datum_des_fehlertupels->#_30042000].

#k_5965_a_fehlertupel_oelfilter_verschmutzt_2:#fehlertupel[
        #hat_bedeutung->2.0;
        #hat_auftrittswahrscheinlichkeit->1.0;
        #hat_entdeckungswahrscheinlichkeit->1.0;
        #hat_massnahme->>#k_5965_a_fehlermeldung_6_wurde_programmiert;
        #ist_fehlerursache_zugeordnet->>#k_5965_a_oelfilter_verschmutzt;
        #hat_massnahme->>#k_5965_a_wartungsplan_ueberarbeitet_2;
        #ist_verantwortlich->>#arndt;
        #datum_des_fehlertupels->#_03062000].

#k_5965_a_fehlertupel_system_zur_pde_wurde_abgestellt_2:#fehlertupel[
        #hat_bedeutung->3.0;
        #hat_auftrittswahrscheinlichkeit->3.0;
        #hat_entdeckungswahrscheinlichkeit->1.0;
        #ist_fehlerursache_zugeordnet->>#k_5965_a_system_zur_pde_wurde_abgestellt;
        #hat_massnahme->>#k_5965_a_system_kann_nur_kontrolliert_abgestellt_werden;
        #ist_verantwortlich->>#ksm_meier;
        #datum_des_fehlertupels->#_03062000].

#k_5965_b_fehlertupel_getriebe_ist_in_gekippter_stellung:#fehlertupel[
        #hat_bedeutung->2.0;
        #hat_auftrittswahrscheinlichkeit->1.0;
        #hat_entdeckungswahrscheinlichkeit->1.0;
        #ist_fehlerursache_zugeordnet->>#k_5965_b_getriebe_ist_in_gekippter_stellung;
        #hat_massnahme->>#k_5965_b_fehlermeldung_1_wurde_programmiert].
```

```
#k_5965_b_fehlertupel_keine_bauteile_montiert_1:#fehlertupel[
        #hat_bedeutung->2.0;
        #hat_auftrittswahrscheinlichkeit->4.0;
        #hat_entdeckungswahrscheinlichkeit->4.0;
        #ist_fehlerursache_zugeordnet->>#k_5965_b_keine_bauteile_montiert;
        #datum_des_fehlertupels->#_30042000].

#k_5965_b_fehlertupel_reflektor_defekt_2:#fehlertupel[
        #hat_bedeutung->3.0;
        #hat_auftrittswahrscheinlichkeit->1.0;
        #hat_entdeckungswahrscheinlichkeit->1.0;
        #ist_fehlerursache_zugeordnet->>#k_5965_b_reflektor_defekt;
        #hat_massnahme->>#k_5965_b_fehlermeldung_14_wurde_programmiert;
        #hat_massnahme->>#k_5965_b_wartungsplan_ueberarbeitet_1;
        #ist_verantwortlich->>#arndt;
        #datum_des_fehlertupels->#_03062000].

#k_5965_b_fehlertupel_taste_getriebe_nicht_in_ordnung_gedrueckt:#fehlertupel[
        #hat_bedeutung->3.0;
        #hat_auftrittswahrscheinlichkeit->1.0;
        #hat_entdeckungswahrscheinlichkeit->1.0;
       #ist_fehlerursache_zugeordnet->>#k_5965_b_taste_getriebe_nicht_in_ordnung_gedrueckt].

#k_5965_b_fehlertupel_passfeder_der_kupplung_abgeschert:#fehlertupel[
        #hat_bedeutung->4.0;
        #hat_auftrittswahrscheinlichkeit->1.0;
        #hat_entdeckungswahrscheinlichkeit->1.0;
        #ist_fehlerursache_zugeordnet->>#k_5965_b_passfeder_der_kupplung_abgeschert].

#k_5965_b_fehlertupel_friktion_der_rollen_zu_schwach_eingestellt:#fehlertupel[
        #hat_bedeutung->2.0;
        #hat_auftrittswahrscheinlichkeit->1.0;
        #hat_entdeckungswahrscheinlichkeit->1.0;
       #ist_fehlerursache_zugeordnet->>#k_5965_b_friktion_der_rollen_zu_schwach_eingestellt].

#k_5965_b_fehlertupel_kegelrad_verschlissen:#fehlertupel[
        #hat_bedeutung->4.0;
        #hat_auftrittswahrscheinlichkeit->1.0;
        #hat_entdeckungswahrscheinlichkeit->1.0;
        #ist_fehlerursache_zugeordnet->>#k_5965_b_kegelrad_verschlissen].

#k_5965_b_fehlertupel_maschine_1_ist_nicht_im_automatikbetrieb:#fehlertupel[
        #hat_bedeutung->2.0;
        #hat_auftrittswahrscheinlichkeit->1.0;
        #hat_entdeckungswahrscheinlichkeit->1.0;
        #ist_fehlerursache_zugeordnet->>#k_5965_b_maschine_1_nicht_im_automatikbetrieb].

#k_5965_b_fehlertupel_maschine_1_ist_abgeschaltet:#fehlertupel[
        #hat_bedeutung->2.0;
        #hat_auftrittswahrscheinlichkeit->1.0;
        #hat_entdeckungswahrscheinlichkeit->2.0;
    #ist_fehlerursache_zugeordnet->>#k_5965_b_maschine_1_ist_abgeschaltet_und_schluesselschalter].

#k_5965_b_fehlertupel_schalter_wt_in_position_2_defekt:#fehlertupel[
        #hat_bedeutung->3.0;
        #hat_auftrittswahrscheinlichkeit->1.0;
        #hat_entdeckungswahrscheinlichkeit->1.0;
      #ist_fehlerursache_zugeordnet->>#k_5965_b_schalter_werkstuecktraeger_in_position_2_defekt;
        #hat_massnahme->>#k_5965_b_fehlermeldung_1_ueber_laufzeitkontrolle].
```

```
#k_5965_b_fehlertupel_maschine_2_ist_abgeschaltet:#fehlertupel[
                    #hat_bedeutung->2.0;
                    #hat_auftrittswahrscheinlichkeit->1.0;
                    #hat_entdeckungswahrscheinlichkeit->2.0;
            #ist_fehlerursache_zugeordnet->>#k_5965_b_maschine_2_ist_abgeschaltet_und_schluesselschalter].

#k_5965_b_fehlertupel_staudruck_wt_zu_gross:#fehlertupel[
                    #hat_bedeutung->2.0;
                    #hat_auftrittswahrscheinlichkeit->2.0;
                    #hat_entdeckungswahrscheinlichkeit->2.0;
                    #ist_fehlerursache_zugeordnet->>#k_5965_b_staudruck_der_werkstuecktraeger_zu_gross].

#k_5965_b_fehlertupel_schutztuer_wurde_geoeffnet:#fehlertupel[
                    #hat_bedeutung->2.0;
                    #hat_auftrittswahrscheinlichkeit->3.0;
                    #hat_entdeckungswahrscheinlichkeit->2.0;
             #ist_fehlerursache_zugeordnet->>#k_5965_b_schutztuer_wurde_geoeffnet_und_schluesselschalter].



// MASSNAHMEN K 5965 --------------------

#k_5965_a_fehlermeldung_1_wurde_programmiert:#entdeckungsmassnahme[
                    #hat_bezeichnung->"Fehlermeldung (1) wurde programmiert"].

#k_5965_a_system_kann_nur_kontrolliert_abgestellt_werden:#entdeckungsmassnahme[
                    #hat_bezeichnung->"System kann nur kontrolliert abgestellt werden"].

#k_5965_a_wartungsplan_ueberarbeitet_1:#vermeidungsmassnahme[
                    #hat_bezeichnung->"Wartungsplan überarbeitet (Öl nachfüllen)"].

#k_5965_a_wartungsplan_ueberarbeitet_2:#vermeidungsmassnahme[
                    #hat_bezeichnung->"Wartungsplan überarbeitet (Filter austauschen)"].

#k_5965_b_fehlermeldung_2_wurde_programmiert:#entdeckungsmassnahme[
                    #hat_bezeichnung->"Fehlermeldung (2) wurde programmiert"].

#k_5965_b_fehlermeldung_1_ueber_laufzeitkontrolle:#entdeckungsmassnahme[
                    #hat_bezeichnung->"Fehlermeldung (1) über Laufzeitkontrolle"].

#k_5965_b_wartungsplan_ueberarbeitet_1:#vermeidungsmassnahme[
                    #hat_bezeichnung->"Wartungsplan überarbeitet"].



// VERANTWORTUNGSTRÄGER-------
// PERSONEN --------------------------------

#ksm_mueller:#person[
                    #hat_name->"Müller";
                    #arbeitet_fuer->>#ksm].

#ksm_meier:#person[
                    #hat_name->"Meier";
                    #arbeitet_fuer->>#ksm].

#ksm_scholz:#person[
                    #hat_name->"Scholz";
                    #arbeitet_fuer->>#ksm].
```

```
// UNTERNEHMEN --------------------

#ksm:#unternehmen[
                    #hat_name->"Karl Schumacher Maschinenbau GmbH"].

#arndt:#unternehmen[
                    #hat_name->"Arndt AG"].



// DATEN --------------------

#_30042000:#datum[
          #jahr->2000;
          #monat->4;
          #tag->30].

#_03062000:#datum[
          #jahr->2000;
          #monat->6;
          #tag->3].

#_23072004:#datum[
          #jahr->2004;
          #monat->7;
          #tag->23].



// REGELN --------------------

// Diese Regeln legen die Symmetriebeziehungen von Relationen fest.

FORALL X,Y
          X[#untersucht_systemelement->Y] <-> Y[#wird_untersucht_von_fmea->X].

FORALL X,Y
          X[#ist_teil_von_systemelement->>Y] <-> Y[#ist_zusammengesetzt_aus_systemelement->>X].

FORALL X,Y
          X[#ist_teil_von_funktion->>Y] <-> Y[#ist_zusammengesetzt_aus_funktion->>X].

FORALL X,Y
          X[#hat_funktion->>Y] <-> Y[#ist_funktion_von_systemelement->>X].

FORALL X,Y
          X[#hat_ursache->>Y] <-> Y[#verursacht->>X].

FORALL X,Y
          X[#hat_fehlermoeglichkeit->>Y] <-> Y[#verhindert_funktion->>X].

FORALL X,Y
          X[#hat_fehlertupel->>Y] <-> Y[#ist_fehlerursache_zugeordnet->>X].

FORALL X,Y
          X[#hat_massnahme->>Y] <-> Y[#wird_angewendet->>X].

FORALL X,Y
          X[#ist_verantwortlich->>Y] <-> Y[#ist_verantwortlich_fuer_fehlertupel->>X].
```

// Die beiden folgenden Regeln verknüpfen über die Relation #ist_teilfunktion diejenigen Systemelemente
// mit einer Funktion, die entweder direkt diese Funktion verwenden oder aus einem Systemelement bestehen,
// das diese Funktion verwendet. Diese Regeln werden zur Zuordnung von Funktionen zu einer FMEA benö
// tigt.

```
FORALL X,Y
        X[#ist_teilfunktion->>Y] <- X[#ist_funktion_von_systemelement->>Y].

FORALL X,Y,Z
        X[#ist_teilfunktion->>Y] <- X[#ist_funktion_von_systemelement->>Z] AND
                                    Z[#ist_teil_von_systemelement->>Y].
```

// Die beiden folgenden Regeln verknüpfen Systemelemente mit den Kopfzeilen einer FMEA.

```
FORALL X,Y
        X[#teil_wird_untersucht_von_fmea->>Y] <- X[#wird_untersucht_von_fmea->Y].

FORALL X,Y,Z
        X[#teil_wird_untersucht_von_fmea->>Y] <- X[#ist_teil_von_systemelement->>Z] AND
                                                 Z[#teil_wird_untersucht_von_fmea->>Y].
```

// Mit diesen folgenden Regeln werden über Built-Ins die Risikoprioritätszahl eines Fehlertupels und die
// Gesamtrisikoprioritätszahl eines Fehlers ermittelt.

```
FORALL W,X,Y,Z,RPZ
        Z:#fehlertupel[#hat_rpz->RPZ] <- Z[#hat_bedeutung->W] AND
                                         Z[#hat_auftrittswahrscheinlichkeit->X] AND
                                         Z[#hat_entdeckungswahrscheinlichkeit->Y] AND
                                         evaluable_(RPZ,*(*(W,X),Y) ).

FORALL X,Y,Z,RPZ,GRPZ
        Y[#hat_gesamtrpz->GRPZ] <- X:#fehlertupel[#hat_rpz->RPZ;
                                           #ist_fehlerursache_zugeordnet->>Z] AND
                                   Y[#hat_ursache->>Z] AND
                                   sum(Y,RPZ,GRPZ).
```

// Die beiden Regeln werden genutzt, um direkte Beziehungen von Konzepten zu Instanzen zu identifizieren.
// Die Relation #is_direct_instance liefert zu einem Konzept die direkt zugeordneten Instanzen.
// Die Relation #is_direct_sub liefert zu einem Konzept die direkt zugeordneten Unterkonzepte. Die Inferenz-
// maschine OntoBroker unterstützt über Built-Ins die Regeln über die Formulierung directisa_(X,Y).
// In der folgenden Kodierung zu möglichen Unterkonzepten wird die kürzere Notation des Built-Ins
// verwendet.

```
FORALL X,Y
        X[#is_direct_instance->>Y] <- (X:Y AND NOT (EXISTS Z((X:Z AND Z::Y)))).

FORALL X,Y
        X[#is_direct_sub->>Y] <- (X::Y AND NOT (EXISTS Z((Z::Y AND X::Z)))).
```

// Die Relation #hat_moegliches_unterkonzept verknüpft mögliche Unterkonzepte mit einem Systemelement.
// Als Beispiele können zugehörige Instanzen mit dem Aufruf directisa_(X,Y) ermittelt werden.

```
FORALL V,W,X,Y,Z
        W:#systemelement[#hat_moegliches_unterkonzept=>>V] <- directisa_(W,X) AND
                                                              directisa_(Y,X) AND
                                          Y[#ist_zusammengesetzt_aus_systemelement->>Z] AND
                                                              directisa_(Z,V).
```

```
// Relation #hat_moegliches_funktionskonzept verbindet mögliche Funktionen mit einem Systemelement.
// Als Beispiel können zugehörige Instanzen mit dem Aufruf des Tupels directisa_(X,Y) ermittelt werden.
// Die Relation #hat_moegliche_funktionsbezeichnung verknüpft mögliche Funktionsbezeichnungen mit
// Systemelementen.

FORALL U,V,W,X,Y,Z
    V:#systemelement[#hat_moegliches_funktionskonzept=>>U;
                     #hat_moegliche_funktionsbezeichnung->Z] <- directisa_(V,W) AND
                                                                directisa_(X,W) AND
                                                                X[#hat_funktion->>Y] AND
                                                                Y[#hat_funktionsbezeichnung->Z] AND
                                                                directisa_(Y,U).
// Die Relation #hat_moeglichen_fehler verknüpft mögliche Fehler mit einer Funktion.

FORALL V,W,X,Y,Z
    W:#funktion[#hat_moeglichen_fehler=>>V] <- W[#ist_funktion_von_systemelement->>Z] AND
                                               directisa_(W,X) AND
                                               V[#verhindert_funktionskonzept=>>X] AND
                                               directisa_(Z,Y) AND
                                               V[#stoert_systemelementkonzept=>>Y].


// Die Relation #verhindert_funktionskonzept verknüpft Konzepte einer Funktion mit einem Fehler,
// deren Instanzen von dem Fehler verhindert werden.
// Die Konzepte werden zur Kategorisierung eines Fehlers verwendet: Die Kategorie eines Fehlers
// ergibt sich aus der Kombination des Konzepts der Funktion, deren Erfüllung er verhindert, mit dem Kon-
// zept des Systemelements, das durch den Fehler gestört wird.

FORALL X,Y,Z
    Y:#fehler[#verhindert_funktionskonzept=>>X] <- Y[#verhindert_funktion->>Z] AND
                                                   directisa_(Z,X).


// Die Relation #stoert_systemelementkonzept verknüpft Konzepte eines Systemelements mit einem
// Fehler, deren Instanzen von dem Fehler gestört werden.
// Die Konzepte werden zur Kategorisierung eines Fehlers verwendet: Die Kategorie eines Fehlers
// ergibt sich aus der Kombination des Konzepts der Funktion, deren Erfüllung er verhindert, mit dem Kon-
// zept des Systemelements, das durch den Fehler gestört wird.

FORALL W,X,Y,Z
    X:#fehler[#stoert_systemelementkonzept=>>W] <- X[#verhindert_funktion->>Y] AND
                                                   Y[#ist_funktion_von_systemelement->>Z] AND
                                                   directisa_(Z,W).

</ns>
```

A.2.2 Erweiterung Wissensbasis durch C 5960 A

```
#c_5960_a_fmea:#fmea[
        #ist_system_fmea_produkt->"true";
        #hat_fmea_nr->"C 5960 A";
        #betrachtet_system_von_typ_modell_fertigung_charge->"9+1 Spindelschrauber";
        #hat_verantwortung->>#ksm_mueller;
        #wurde_erstellt->#_03062000].

#ksm_mueller[
        arbeitet_fuer->>#ksm].

#c_5960_a_neun_eins_spindelschrauber:#schraubmaschine[
        #hat_systemelementbezeichnung->"neun eins spindelschrauber";
        #wird_untersucht_von_fmea->#c_5960_a_fmea].

#c_5960_a_bedienpult:#bedieneinheit[
        #hat_systemelementbezeichnung->"Bedienpult";
        #ist_teil_von_systemelement->>#c_5960_a_neun_eins_spindelschrauber].

#c_5960_a_verschraubt_getriebe:#fixieren[
        #hat_funktionsbezeichnung->"Verschraubt Getriebe";
        #ist_funktion_von_systemelement->>#c_5960_a_neun_eins_spindelschrauber].

#c_5960_a_umschaltung_automatik_oder_manuellbetrieb:#schalten[
        #hat_funktionsbezeichnung->"Umschaltung Automatik- oder Manuellbetrieb";
        #ist_funktion_von_systemelement->>#c_5960_a_bedienpult].

#c_5960_a_umschaltung_automatik_oder_manuellbetrieb[
        #ist_teil_von_funktion->>#c_5960_a_verschraubt_getriebe].

#c_5960_a_maschinenstoerung:#fehler[
        #verhindert_funktion->>#c_5960_a_verschraubt_getriebe;
        #hat_fehlerbezeichnung->"Maschinenstoerung"].

#c_5960_a_anwahl_automatik_und_handbetrieb_nicht_moeglich:#fehler[
        #verhindert_funktion->>#c_5960_a_umschaltung_automatik_oder_manuellbetrieb;
        #hat_fehlerbezeichnung->"Anwahl Automatik und Handbetrieb nicht moeglich"].

#c_5960_a_anwahl_automatik_nicht_moeglich:#fehler[
        #verhindert_funktion->>#c_5960_a_umschaltung_automatik_oder_manuellbetrieb;
        #hat_fehlerbezeichnung->"Anwahl Automatik nicht moeglich"].

#c_5960_a_pneumatikversorgung_fehlt:#fehler[
        #verhindert_funktion->>#c_5960_a_umschaltung_automatik_oder_manuellbetrieb;
        #hat_fehlerbezeichnung->"Pneumatikversorgung fehlt"].

#c_5960_a_druckschalter_pneumatik_meldet_nicht:#fehler[
        #verhindert_funktion->>#c_5960_a_umschaltung_automatik_oder_manuellbetrieb;
        #hat_fehlerbezeichnung->"Druckschalter Pneumatik meldet nicht"].

#c_5960_a_schutztuerschalter_defekt:#fehler[
        #verhindert_funktion->>#c_5960_a_umschaltung_automatik_oder_manuellbetrieb;
        #hat_fehlerbezeichnung->"Schutztuerschalter defekt"].

#c_5960_a_maschine_atlas_copco_nicht_betriebsbereit:#fehler[
        #verhindert_funktion->>#c_5960_a_umschaltung_automatik_oder_manuellbetrieb;
        #hat_fehlerbezeichnung->"Maschine (Atlas Copco) nicht betriebsbereit"].
```

```
#c_5960_a_maschine_nicht_in_grundstellung:#fehler[
        #verhindert_funktion->>#c_5960_a_umschaltung_automatik_oder_manuellbetrieb;
        #hat_fehlerbezeichnung->"Maschine nicht in Grundstellung"].

#c_5960_a_anwahl_automatik_und_handbetrieb_nicht_moeglich:#fehler[
        #verursacht->>#c_5960_a_maschinenstoerung].

#c_5960_a_anwahl_automatik_nicht_moeglich:#fehler[
        #verursacht->>#c_5960_a_maschinenstoerung].

#c_5960_a_pneumatikversorgung_fehlt:#fehler[
        #verursacht->>#c_5960_a_anwahl_automatik_und_handbetrieb_nicht_moeglich].

#c_5960_a_druckschalter_pneumatik_meldet_nicht:#fehler[
        #verursacht->>#c_5960_a_anwahl_automatik_und_handbetrieb_nicht_moeglich].

#c_5960_a_schutztuerschalter_defekt:#fehler[
        #verursacht->>#c_5960_a_anwahl_automatik_und_handbetrieb_nicht_moeglich].

#c_5960_a_maschine_atlas_copco_nicht_betriebsbereit:#fehler[
        #verursacht->>#c_5960_a_anwahl_automatik_nicht_moeglich].

#c_5960_a_maschine_nicht_in_grundstellung:#fehler[
        #verursacht->>#c_5960_a_anwahl_automatik_nicht_moeglich].

#c_5960_a_fehlermeldung_1_wurde_programmiert:#entdeckungsmassnahme[
        #hat_bezeichnung->"Fehlermeldung (1) wurde programmiert"].

#c_5960_a_fehlermeldung_3_wurde_programmiert:#entdeckungsmassnahme[
        #hat_bezeichnung->"Fehlermeldung (3) wurde programmiert"].

#c_5960_a_fehlermeldung_3_wurde_programmiert_1:#entdeckungsmassnahme[
        #hat_bezeichnung->"Fehlermeldung (3) wurde programmiert"].

#c_5960_a_fehlertupel_schutztuerschalter_defekt:#fehlertupel[
        #ist_fehlerursache_zugeordnet->>#c_5960_a_schutztuerschalter_defekt;
        #hat_massnahme->>#c_5960_a_fehlermeldung_3_wurde_programmiert_1;
        #datum_des_fehlertupels->#_03062000].

#c_5960_a_fehlermeldung_1_wurde_programmiert_1:#entdeckungsmassnahme[
        #hat_bezeichnung->"Fehlermeldung (1) wurde programmiert"].

#c_5960_a_fehlertupel_druckschalter_pneumatik_meldet_nicht:#fehlertupel[
        #ist_fehlerursache_zugeordnet->>#c_5960_a_druckschalter_pneumatik_meldet_nicht;
        #hat_massnahme->>#c_5960_a_fehlermeldung_1_wurde_programmiert_1;
        #datum_des_fehlertupels->#_03062000].

#c_5960_a_fehlermeldung_4_wurde_programmiert:#entdeckungsmassnahme[
        #hat_bezeichnung->"Fehlermeldung (4) wurde programmiert"].

#c_5960_a_fehlertupel_maschine_atlas_copco_nicht_betriebsbereit:#fehlertupel[
    #ist_fehlerursache_zugeordnet->>#c_5960_a_maschine_atlas_copco_nicht_betriebsbereit;
    #hat_massnahme->>#c_5960_a_fehlermeldung_4_wurde_programmiert;
    #datum_des_fehlertupels->#_03062000].

#c_5960_a_fehlermeldung_5_wurde_programmiert:#entdeckungsmassnahme[
    #hat_bezeichnung->"Fehlermeldung (5) wurde programmiert"].

#c_5960_a_fehlertupel_maschine_nicht_in_grundstellung:#fehlertupel[
    #ist_fehlerursache_zugeordnet->>#c_5960_a_maschine_nicht_in_grundstellung;
    #hat_massnahme->>#c_5960_a_fehlermeldung_5_wurde_programmiert;
    #datum_des_fehlertupels->#_03062000].
```

```
#c_5960_a_fehlermeldung_2_wurde_programmiert:#entdeckungsmassnahme[
        #hat_bezeichnung->"Fehlermeldung (2) wurde programmiert"].

#c_5960_a_fehlertupel_pneumatikversorgung_fehlt:#fehlertupel[
        #ist_fehlerursache_zugeordnet->>#c_5960_a_pneumatikversorgung_fehlt;
        #hat_massnahme->>#c_5960_a_fehlermeldung_2_wurde_programmiert;
        #datum_des_fehlertupels->#_03062000].

#c_5960_a_fehlertupel_schutztuerschalter_defekt[
        #hat_bedeutung->2;
        #hat_auftrittswahrscheinlichkeit->1;
        #hat_entdeckungswahrscheinlichkeit->1].

#c_5960_a_fehlertupel_pneumatikversorgung_fehlt[
        #hat_bedeutung->2;
        #hat_auftrittswahrscheinlichkeit->1;
        #hat_entdeckungswahrscheinlichkeit->1].

#c_5960_a_fehlertupel_maschine_atlas_copco_nicht_betriebsbereit[
        #hat_bedeutung->2;
        #hat_auftrittswahrscheinlichkeit->1;
        #hat_entdeckungswahrscheinlichkeit->1].

#c_5960_a_fehlertupel_maschine_nicht_in_grundstellung[
        #hat_bedeutung->2;
        #hat_auftrittswahrscheinlichkeit->1;
        #hat_entdeckungswahrscheinlichkeit->1].

#c_5960_a_fehlertupel_druckschalter_pneumatik_meldet_nicht[
        #hat_bedeutung->2;
        #hat_auftrittswahrscheinlichkeit->1;
        #hat_entdeckungswahrscheinlichkeit->1].
```